AFRL-PR-WP-TR-1999-2028

THE INTEGRATED MULTI-OBJECTIVE MULTI-DISCIPLINARY JET ENGINE DESIGN OPTIMIZATION PROGRAM

NICHOLAS J. KUPROWICZ

AIR FORCE INSTITUTE OF TECHNOLOGY
2950 P STREET
WRIGHT-PATTERSON AFB, OH 45433-7765

JANUARY 1999

FINAL REPORT FOR MARCH 1997 - DECEMBER 1998

PROPULSION DIRECTORATE
AIR FORCE RESEARCH LABORATORY
AIR FORCE MATERIEL COMMAND
WRIGHT-PATTERSON AIR FORCE BASE OH 45433-7251

[illegible]
ERROL G. BLEVINS
Project Engineer
Engine Integration & Assessment Branch
Turbine Engine Division
Propulsion Directorate

[illegible]
RICHARD J. KRABAL
Chief
Engine Integration & Assessment Branch
Turbine Engine Division
Propulsion Directorate

William E Koop
WILLIAM E. KOOP
Chief of Technology
Turbine Engine Division
Propulsion Directorate

REPORT DOCUMENTATION PAGE			*Form Approved* *OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** JANUARY 1999	**3. REPORT TYPE AND DATES COVERED** FINAL REPORT FOR MAR 1997 - DEC 1998	
4. TITLE AND SUBTITLE THE INTEGRATED MULTI-OBJECTIVE MULTI-DISCIPLINARY JET ENGINE DESIGN OPTIMIZATION PROGRAM			**5. FUNDING NUMBERS** IN- HOUSE PE 62203 PR 3066 TA 11 WU TC
6. AUTHOR(S) NICHOLAS J. KUPROWICZ			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) AIR FORCE INSTITUTE OF TECHNOLOGY 2950 P STREET WRIGHT-PATTERSON AFB, OH 45433-7765			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) PROPULSION DIRECTORATE AIR FORCE RESEARCH LABORATORY AIR FORCE MATERIEL COMMAND WRIGHT-PATTERSON AFB, OH 45433-7251 POC: ERROL G. BLEVINS, AFRL/PRTA, 937-255-5308			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER** AFRL-PR-WP-TR-1999-2028
11. SUPPLEMENTARY NOTES			
12a. DISTRIBUTION AVAILABILITY STATEMENT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED			**12b. DISTRIBUTION CODE**

13. ABSTRACT *(Maximum 200 words)*

The integrated multi-objective multi-disciplinary jet engine design optimization program is an analysis tool to aid engineers in the conceptual engine design process. The program allows performance evaluation of a specified engine or a specified aircraft/engine combination at given operating conditions or over a given mission. In addition, the program allows the selection of values for specified engine parameters that yield the best composite performance at one or more operating conditions or over a given mission. Finally, the program utilizes multi-objective optimization techniques to simultaneously address conflicting objectives such as maximizing performance and minimizing fuel use, size, and cost.

This report is primarily a software user's guide to provide instruction on using the Integrated Multi-Objective Multi-Disciplinary Jet Engine Design Optimization Program. The genetic algorithm routines used in the program are based on an existing public-domain package. The aircraft design program is based on a AIAA sponsored code. The engine performance program is a proprietary DOD-limited code.

14. SUBJECT TERMS Jet Engine Optimization, Micro-Genetic Algorithm, Genetic Algorithm, Non-Linear Optimization, Estimation, Kriging			**15. NUMBER OF PAGES** 81
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	**18. SECURITY CLASSIFICATION OF THIS PAGE** UNCLASSIFIED	**19. SECURITY CLASSIFICATION OF ABSTRACT** UNCLASSIFIED	**20. LIMITATION OF ABSTRACT** SAR

Standard Form 298 (Rev. 2-89) (EG)
Prescribed by ANSI Std. 239.18
Designed using Perform Pro, WHS/DIOR, Oct 94

Contents

List of Figures

List of Tables

Executive Summary

The Integrated Multi-Objective Multi-Disciplinary Jet Engine Design Optimization Program is an analysis tool to aid engineers in the conceptual jet engine design process. Specifically, it allows

- Performance evaluation of a specified engine at given operating conditions
- Performance evaluation of a specified aircraft/engine combination over a given mission
- Selection of values for specified engine parameters that yield the best composite performance at one or more operating conditions
- Selection of values for specified engine parameters that, when integrated with the aircraft, yield the best composite performance over a given mission
- Optional inclusion of non-constant installation losses during evaluation
- Optional ability to size aircraft while optimizing over a mission
- Ability to tailor optimization by modifying all pertinent optimization control parameters
- Ability to reduce total run time by including adaptive function estimation (kriging)

Engine cycle analysis is performed using TERMAP [1] , a sophisticated analysis code developed by Allison ADC under the direction of the USAF Foreign Technology Division.

Acronyms

The acronyms used throughout this document are given below.

A/B Afterburner

A/C Aircraft

GA Genetic Algorithm

GAOT Genetic Algorithms for Optimization Toolbox

GUI Graphical User Interface

I/O Input/Output

OS Operating System

RAM Random Access Memory

ROM Read Only Memory

SDD Software Design Document

[1]The Turbine Engine Reverse Modeling Aid Program (TERMAP) is a proprietary, DoD-Limited computer program. Only that information which has been approved for public release is included in this document.

SUG Software User's Guide

S/W Software

TERMAP Turbine Engine Reverse Modeling Aid Program

References

1. Nadon, Luc J. J. P., *Multidisciplinary and Multiobjective Optimization in Conceptual Design for Mixed-Stream Turbofan Engines*, MS Thesis, AFIT/GAE/ENY/96D-6, Air Force Institute of Technology, Wright-Patterson AFB, OH, 1996

2. Millhouse, Paul T., *Improving Algorithmic Efficiency of Aircraft Engine Design for Optimal Performance*, MS Thesis, AFIT/GOR/ENY/98M-02, Air Force Institute of Technology, Wright Patterson AFB, OH, 1998

3. Mattingly, Jack D., Heiser, William H., Daley, Daniel H., *Aircraft Engine Design*, AIAA Education Series, AIAA, New York, 1987

4. Hill, Philip G., Peterson, Carl R., *Mechanics and Thermodynamics of Propulsion*, 2nd Edition, Addison-Wesley Publishing Co., Massachusetts, 1992

5. The MathWorks, Inc., *Using MATLAB: Version 5*, The MathWorks, Inc., 1996

6. The MathWorks, Inc., *Using MATLAB Graphics: Version 5*, The MathWorks, Inc., 1996

7. Allison Advanced Development Company, *TERMAP Program User's Manual, Vol. 1* [2]

8. Houck, Christopher R., Joines, Jeffery A., Kay, Michael G., *A Genetic Algorithm for Function Optimization: A Matlab Implementation*, contained with the Matlab GAOT toolbox

[2]This reference is not public.

Chapter 1

Software User's Guide

1.1 Overview

The purpose of this Software User's Guide (SUG) is to provide instruction on using the Integrated Multi-Objective Multi-Disciplinary Jet Engine Design Optimization Program. Details concerning software design are covered in the SDD.

1.1.1 Hardware and Software Requirements

Both UNIX and PC Windows (Windows 95/98/NT) operating systems are supported by the program.

UNIX

On machines with the UNIX OS, it is required that Matlab version 5.1 or higher be installed. The machine must have a minimum 128 Mb RAM, and a minimum 266 MHz processor. It is highly recommended that a 17-inch or larger monitor be used.

It is also required that the user furnish a UNIX executable version of TERMAP which has had the interfacing modifications outlined in the SDD. *A UNIX executable version of TERMAP which has had these modifications, along with complete FORTRAN-77 source code, is contained on a DoD-Limited CD-ROM separate from this document.*

Should the user desire the capability to directly access a text editor from within the program, it is also required that an executable version of a text editor reside on the UNIX machine.

PC Windows

On machines with a PC Windows (Windows 95/98/NT) OS, it is required that Matlab version 5.2.1 or higher be installed. The machine must have a minimum 128 Mb RAM, and a minimum 400 MHz processor. It is highly recommended that a 17-inch or larger monitor be used.

It is also required that the user furnish a DOS executable version of TERMAP which has had the interfacing modifications outlined in the SDD. *The interface changes required for a DOS version of TERMAP are identical to that required for a UNIX version.*

Should the user desire the capability to directly access a text editor from within the program, it is also required that an executable version of a text editor reside on the PC Windows machine.

1.1.2 Main Functional Areas

The program has been designed around a core set of tasks referred to as the main functional areas. Each of these areas, along with a brief description of their purpose, is discussed in the following sections.

Stand Alone Engine Cycle Analysis

The purpose of this functional area is to evaluate engine performance at specific operating conditions. The basic inputs to this function are an on design engine specification, an off design point, and the variables of interest for analysis. Output data consists of on design and off design performance values for the desired variables.

Stand Alone Mission Analysis

The purpose of this functional area is to evaluate combined aircraft/engine performance for a selected mission. The basic inputs to this function are an on-design engine specification, aircraft characteristics, and a mission profile. Output data consists of a leg by leg performance evaluation.

Engine Optimization with Mission - Fixed A/C

The purpose of this functional area is to determine the engine cycle configuration which optimizes a mission dependent cost functional. The basic inputs to this function are an on-design engine cycle specification (fixed parameters as well as design variables and ranges), aircraft characteristics, a mission profile, design objectives, and optimizer options. Output data consists of the optimizer's best solution along with lower level information describing the optimizer's performance.

Engine Optimization with Mission - Scaleable A/C

The purpose of this functional area is to determine the engine cycle and aircraft size configuration which optimizes a mission dependent cost functional. The basic inputs and outputs of this function are the same as the previous section, except that aircraft sizing parameters are used instead of a fixed takeoff weight value. The use of these sizing parameters results in an additional design variable for optimization runs, and is the reason for treating fixed and scaleable aircraft cases separately.

Engine Optimization without Mission

The purpose of this functional area is to determine the engine cycle configuration which optimizes a cost functional relating a number of operating conditions. The basic inputs to this function are an on-design engine cycle specification (fixed parameters as well as design variables and ranges), a number of off design points, a design objective for each off design point, and optimizer options. Output data consists of the optimizer's best solution along with lower level information describing the optimizer's performance.

1.1.3 Graphical User Interface

The entire program acts as a manipulator of large, structured sets of data referred to as data elements, and a graphical user interface (GUI) has been built into the program which takes advantage of this concept. Each data element is treated as an object, independent of any specific function to be performed. This approach permits data elements to be shared between functional areas, and simplifies the process by which information is stored to and loaded from a disk file.

Each functional area has its own GUI menu in which the user can define the contents of input data elements, execute the core function, and view the contents of output data elements. The layout of each menu is identical, with 4 separate areas on the screen to display and receive user information. The relative positions and sizes of each area are shown in Figure 1.1.

Message Area Error, warning, and other messages to the user are placed in this area.

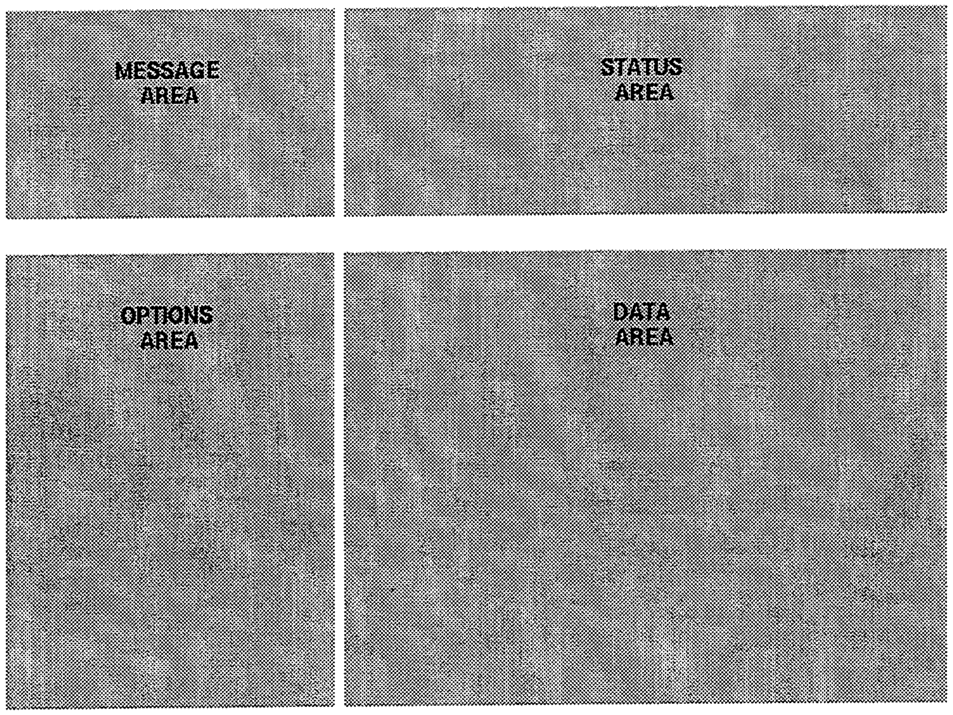


Figure 1.1: Screen Areas

Options Area The options available to the user at any given time are placed in this area.

Status Area Information concerning the status of applicable data elements for the displayed menu is placed in this area.

Data Area The contents of data elements may be viewed and/or edited in this area.

The menus for each of the functional areas are accessible from a base menu which is initially displayed on the screen. There is also an additional menu for performing secondary functions within the program. Before covering further details and procedures, a review of all data elements within the program is provided.

1.2 Data Elements

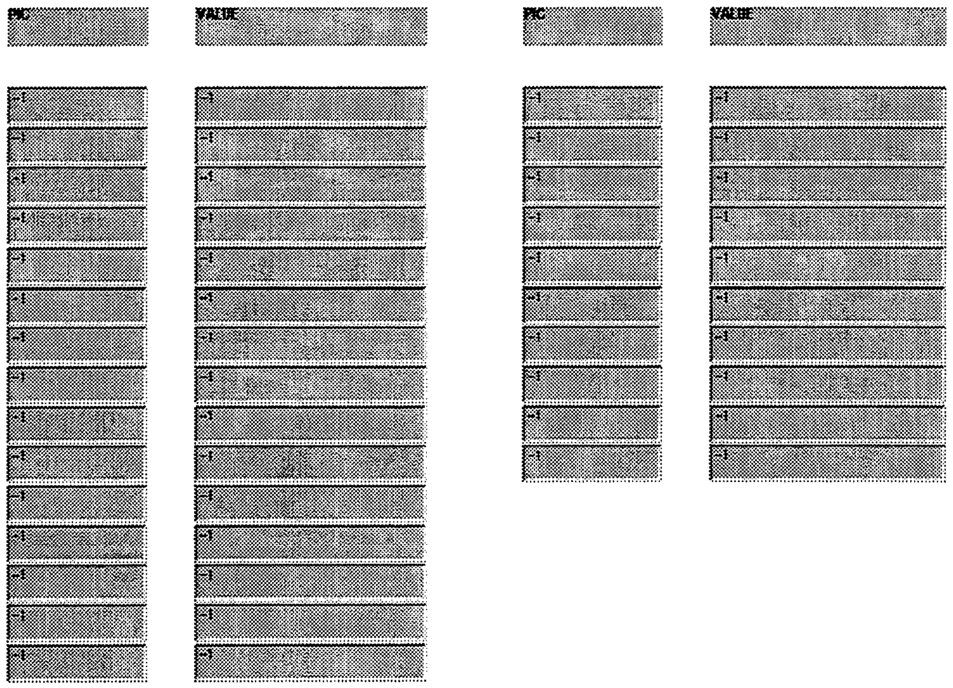
This section describes all data elements used in the program and, where applicable, provides a corresponding image from the data area.

1.2.1 TERMAP Input File - Name and Path

This data element contains the name and path of a baseline TERMAP input file. It is viewed or edited via a standard file selection dialog whose appearance is dependent on the computer system being used. An overview on the use and limitations of a baseline TERMAP input file with the program is given in Section 1.3.1.

1.2.2 Engine On Design Specification

This data element contains a set of PIC/Data pairs which correspond to fixed on design engine parameters. By program design, no more that 25 pairs may be specified. The format of this data element is shown in Figure 1.2.



PIC	VALUE	PIC	VALUE
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1		
-1	-1		
-1	-1		
-1	-1		
-1	-1		

Figure 1.2: Engine On Design Specification

Unused entries must have a PIC value set to -1. Used entries must have an integer PIC value in the range from 1 to 2288.

1.2.3 Engine Maximum Limits

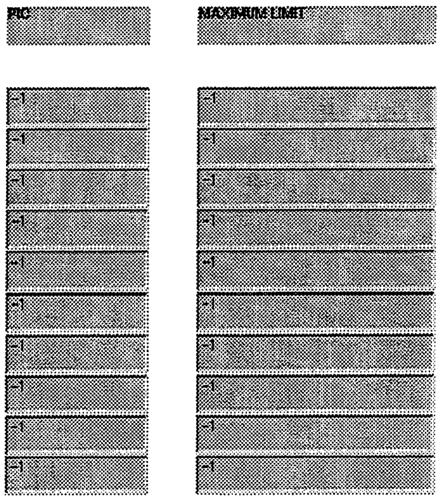
This data element contains a set of PIC/Data pairs which correspond to maximum engine limits. By TERMAP design, no more than 10 maximum limits may be specified. The format of this data element is shown in Figure 1.3.

Unused entries must have a PIC value set to -1. Used entries must have an integer PIC value in the range from 1 to 2288.

1.2.4 Engine Minimum Limits

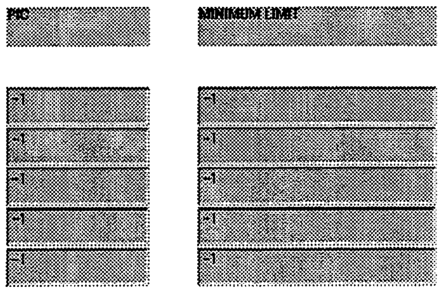
This data element contains a set of PIC/Data pairs which correspond to minimum engine limits. By TERMAP design, no more than 5 minimum limits may be specified. The format of this data element is shown in Figure 1.4.

Unused entries must have a PIC value set to -1. Used entries must have an integer PIC value in the range from 1 to 2288.



PIC	MAXIMUM LIMIT
-1	-1
-1	-1
-1	-1
-1	-1
-1	-1
-1	-1
-1	-1
-1	-1
-1	-1
-1	-1

Figure 1.3: Engine Maximum Limits

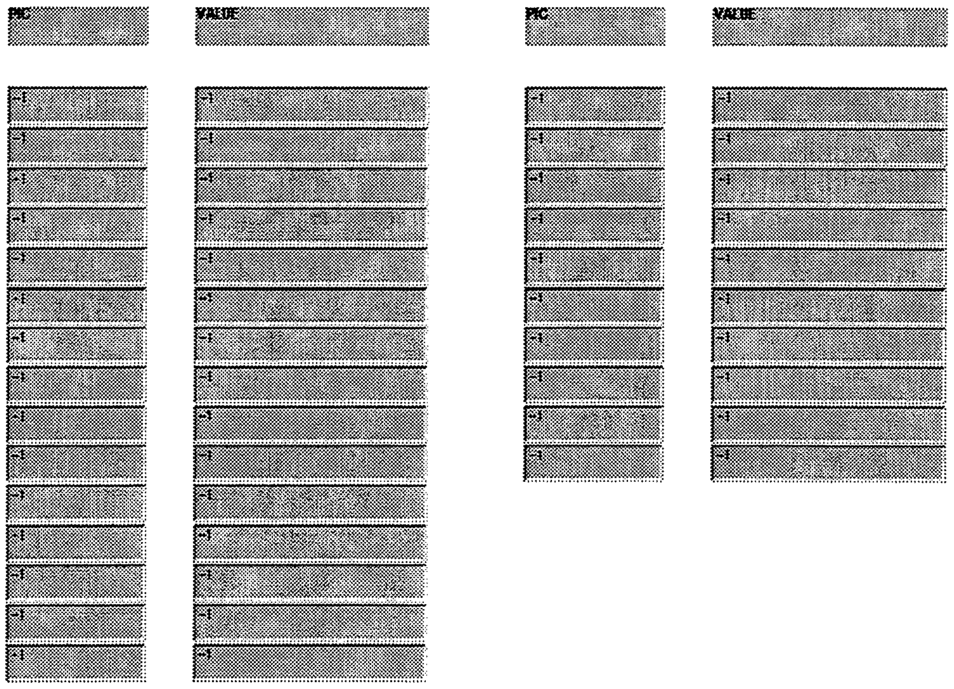


PIC	MINIMUM LIMIT
-1	-1
-1	-1
-1	-1
-1	-1
-1	-1

Figure 1.4: Engine Minimum Limits

1.2.5 Off Design Point Definition

This data element contains a set of PIC/Data pairs which correspond to an off design point. By program design, no more than 25 pairs may be specified. The format of this data element is shown in Figure 1.5.



PIC	VALUE	PIC	VALUE
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1		
-1	-1		
-1	-1		
-1	-1		
-1	-1		

Figure 1.5: Off Design Point Definition

Unused entries must have a PIC value set to -1. Used entries must have an integer PIC value in the range [1] from 1 to 2288.

1.2.6 Desired Outputs - On Design

This data element contains a set of PICs whose data values will be extracted from TERMAP during on design evaluation. By program design, no more that 20 PICs may be specified. The format of this data element is shown in Figure 1.6.

Unused entries must have a PIC value set to -1. Used entries must have an integer PIC value in the range from 1 to 2288.

1.2.7 Desired Outputs - Off Design

This data element contains a set of PICs whose data values will be extracted from TERMAP during off design evaluation. By program design, no more than 20 PICs may be specified. The format of this data element is shown in Figure 1.7.

[1]The modified version of TERMAP recognizes a PIC of 3000 for the MODE parameter. The user should only use this in the off design point definition for functional areas not involving an aircraft mission.

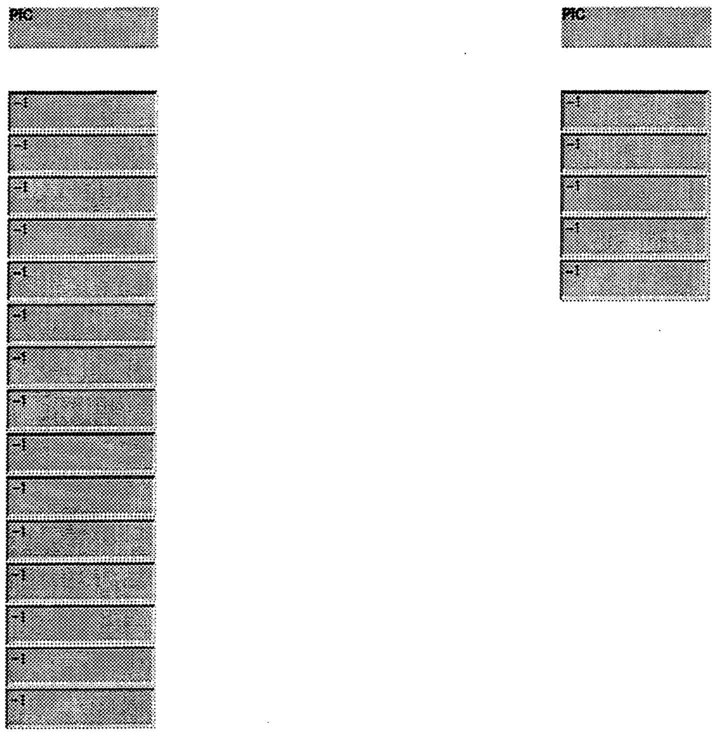


Figure 1.6: Desired Outputs - On Design

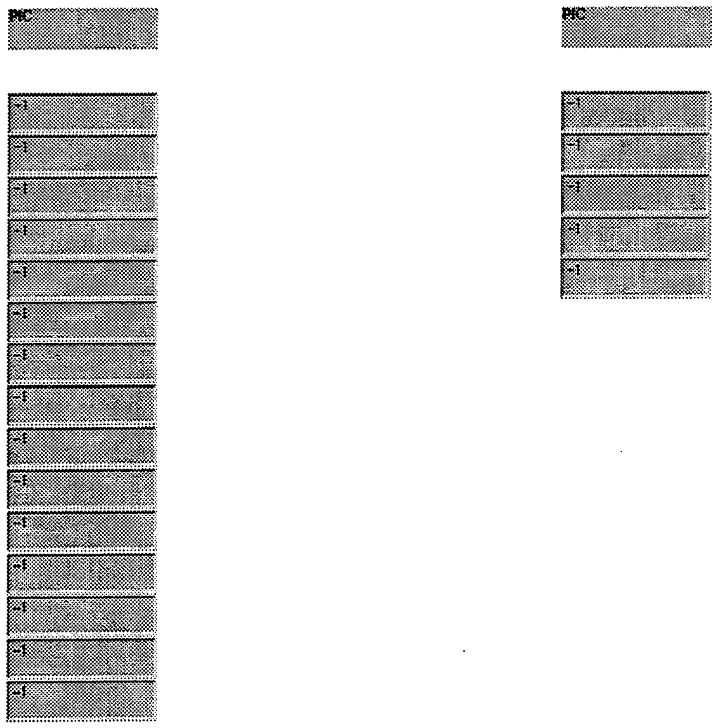


Figure 1.7: Desired Outputs - Off Design

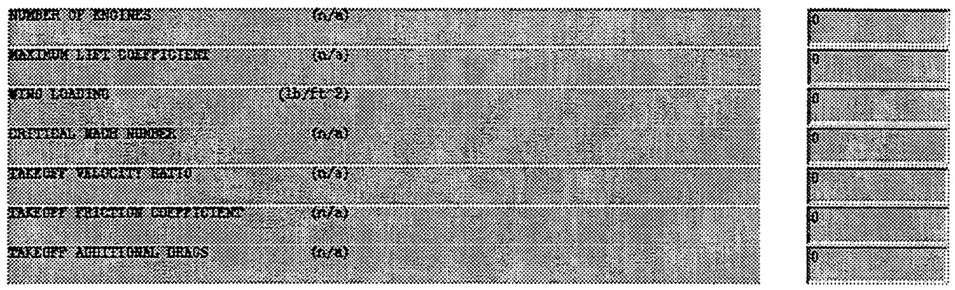
Unused entries must have a PIC value set to -1. Used entries must have an integer PIC value in the range from 1 to 2288.

1.2.8 Aircraft Drag File - Name and Path

This data element contains the name and path of a drag file for an aircraft. It is viewed or edited via a standard file selection dialog whose appearance is dependent on the computer system being used. An overview on the use and limitations of an aircraft drag file with the program is given in Section 1.3.2.

1.2.9 Aircraft Constants

This data element contains various aircraft related constants. The format of this data element is shown in Figure 1.8.



NUMBER OF ENGINES	(n/a)	0
MAXIMUM LIFT COEFFICIENT	(n/a)	0
WING LOADING	(lb/ft^2)	0
CRITICAL MACH NUMBER	(n/a)	0
TAKEOFF VELOCITY RATIO	(n/a)	0
TAKEOFF FRICTION COEFFICIENT	(n/a)	0
TAKEOFF ADDITIONAL DRAGS	(n/a)	0

Figure 1.8: Aircraft Constants

A description of each of these constants, along with their corresponding units, is given in Table 1.1.

Table 1.1: Aircraft Constants

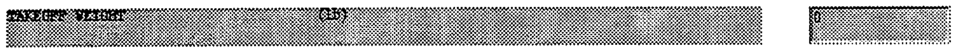
Name	*Units*	*Description*
Number of Engines	n/a	This parameter defines the number of engines installed in an aircraft.
Maximum Lift Coefficient	n/a	This parameter defines the maximum allowable value for the coefficient of lift. During mission evaluation, the calculated value for the lift coefficient is compared to this maximum value. A violation for any mission leg is flagged in the mission output.
Wing Loading	psf	This parameter defines the value of aircraft wing loading. It is used in conjunction with takeoff weight to determine wing surface area.
Critical Mach Number	n/a	This paramter defines the mach number corresponding to a best cruise flight condition. It is used for all mission legs of the best cruise mach type.

Aircraft Constants - continued

Name	*Units*	*Description*
Takeoff Velocity Ratio	n/a	This parameter defines a ratio of required takeoff velocity divided by stall velocity. It is only used for mission legs of the takeoff type.
Takeoff Friction Coefficient	n/a	This parameter defines a coefficient which represents the effect of wheel friction during a takeoff leg evaluation. It is used in the calculation of a total drag coefficient during takeoff.
Takeoff Additional Drags	n/a	This parameter defines a coefficient which represents the effect of landing gear and other sources of drag not accounted for in the A/C drag file during a takeoff leg evaluation. It is used in the calculation of a total drag coefficient during takeoff.

1.2.10 Aircraft Takeoff Weight

This data element contains the gross takeoff weight, in pounds, of a fixed aircraft. The format of this data element is shown in Figure 1.9.

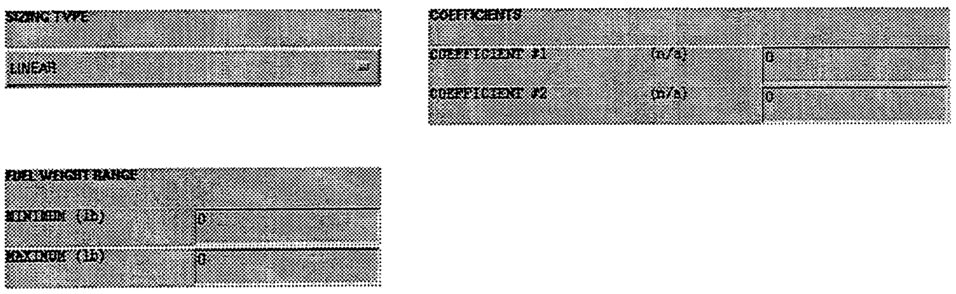


TAKEOFF WEIGHT (lb) 0

Figure 1.9: Aircraft Takeoff Weight

1.2.11 Aircraft Sizing Parameters

This data element contains the parameters necessary for determining the gross takeoff weight of an aircraft as a function of fuel weight. The format of this data element, for a linear sizing scheme, is shown in Figure 1.10.



SIZING TYPE

LINEAR

COEFFICIENTS

COEFFICIENT #1	(n/a)	0
COEFFICIENT #2	(n/a)	0

FUEL WEIGHT RANGE

MINIMUM (lb)	0
MAXIMUM (lb)	0

Figure 1.10: Aircraft Sizing Parameters

Sizing Type

The selected type defines the sizing scheme which will be used for gross takeoff weight calculations. Both linear and nonlinear types are supported.

Fuel Weight Range

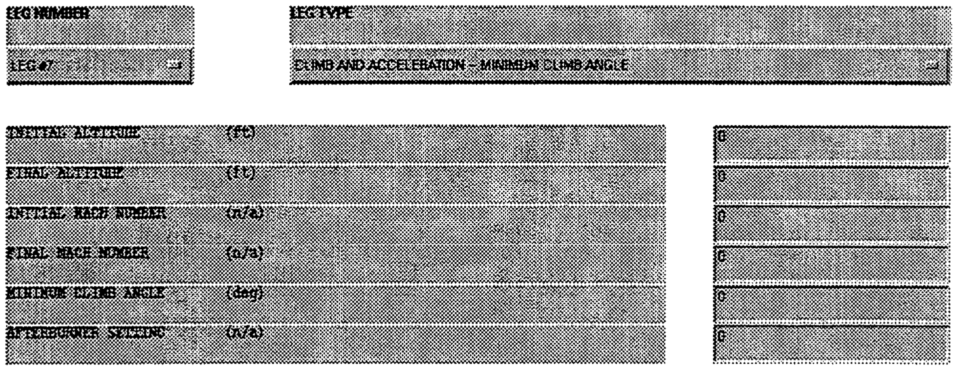
These parameters define a range of fuel weight values which may be used for gross takeoff weight calculations.

Coefficients

These parameters are used in conjunction with a fuel weight value to determine a gross takeoff weight. For a linear scheme, two coefficients are used and gross takeoff weight is calculated as $W_{TO} = C_0 + C_1 W_f$, where W_f denotes fuel weight. For a nonlinear scheme, 10 coefficients are used and gross takeoff weight is calculated as $W_{TO} = C_0 + C_1 W_f + C_2 {W_f}^2 + \ldots + C_9 {W_f}^9$

1.2.12 Mission Profile

This data element contains an aircraft mission, organized by flight leg. The format of this data element, for a specific example, is shown in Figure 1.11.



LEG NUMBER	LEG #7	
LEG TYPE	CLIMB AND ACCELERATION – MINIMUM CLIMB ANGLE	
INITIAL ALTITUDE	(ft)	0
FINAL ALTITUDE	(ft)	0
INITIAL MACH NUMBER	(n/a)	0
FINAL MACH NUMBER	(n/a)	0
MINIMUM CLIMB ANGLE	(deg)	0
AFTERBURNER SETTING	(n/a)	0

Figure 1.11: Mission Profile

Leg Number

The selected value defines a mission leg number. Up to and including 30 mission legs are supported.

Leg Type

The selected type defines, for each leg number, which mission leg calculations will be used during mission evaluation. The names of each leg type, which themselves should provide an adequate description of their function, are given in Table 1.2.

Table 1.2: Leg Types

Type
Unused
Constant Speed Climb - Minimum Climb Angle
Constant Speed Climb - Maximum Distance

Leg Types - continued

Name
Constant Speed Climb - Minimum Climb Rate
Constant Speed Climb - Maximum Time
Horizontal Acceleration - Maximum Distance
Horizontal Acceleration - Maximum Time
Climb and Acceleration - Minimum Climb Angle
Climb and Acceleration - Maximum Distance
Climb and Acceleration - Minimum Climb Rate
Climb and Acceleration - Maximum Time
Takeoff
Constant Altitude/Speed Cruise - Distance
Constant Altitude/Speed Cruise - Time
Constant Altitude/Speed Turn
Best Cruise Mach/Altitude - Distance
Best Cruise Mach/Altitude - Time
Loiter - Distance
Loiter - Time
Warmup
Constant Energy Height Maneuver
Deliver Expendables

Constants

Upon user selection of a leg type, the required constants corresponding to that type are automatically displayed. A description of each of these constants, along with their corresponding units, is given in Table 1.3.

Table 1.3: Leg Constants

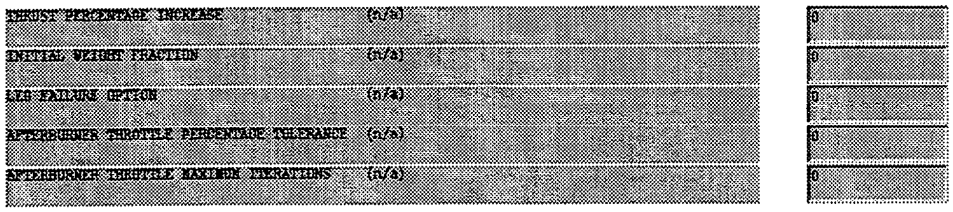
Name	*Units*	*Description*
Altitude	ft	self explanatory
Initial Altitude	ft	self explanatory
Final Altitude	ft	self explanatory
Mach Number	n/a	self explanatory
Initial Mach Number	n/a	self explanatory
Final Mach Number	n/a	self explanatory
Distance	nm	self explanatory
Maximum Distance	nm	self explanatory
Minimum Climb Angle	deg	self explanatory
Minimum Climb Rate	ft/sec	self explanatory
Time	sec	self explanatory
Maximum Time	sec	self explanatory
Rotation Time	sec	self explanatory

Leg Constants - continued

Name	*Units*	*Description*
Afterburner Setting	n/a	This parameter is used for mission legs in which throttling to a thrust is not required. A value of 1 corresponds to max power, while a value of 0 corresponds to mil power. Any value between 0 and 1 may be used.
Afterburner Option	n/a	This parameter is used for mission legs in which throttling to a thrust is required. A value of 1 permits the afterburner to be used (if necessary), while a value of 0 does not permit the afterburner to be used. Only values of 0 and 1 may be used.
Number of Turns	n/a	self explanatory
Load Factor	g	self explanatory
Vertical Fraction	n/a	A value of 0 corresponds to horizontal flight, while a value of 1 corresponds to vertical flight. Any value between 0 and 1 may be used.
Expendable Weight	lb	This parameter corresponds to the weight of an object, such as a weapon, which is released from the aircraft.

1.2.13 Mission Constants

This data element contains various mission related constants. The format of this data element is shown in Figure 1.12. A description of each of these constants, along with their corresponding



THRUST PERCENTAGE INCREASE	(n/a)	0
INITIAL WEIGHT FRACTION	(n/a)	0
LEG FAILURE OPTION	(n/a)	0
AFTERBURNER THROTTLE PERCENTAGE TOLERANCE	(n/a)	0
AFTERBURNER THROTTLE MAXIMUM ITERATIONS	(n/a)	0

Figure 1.12: Mission Constants

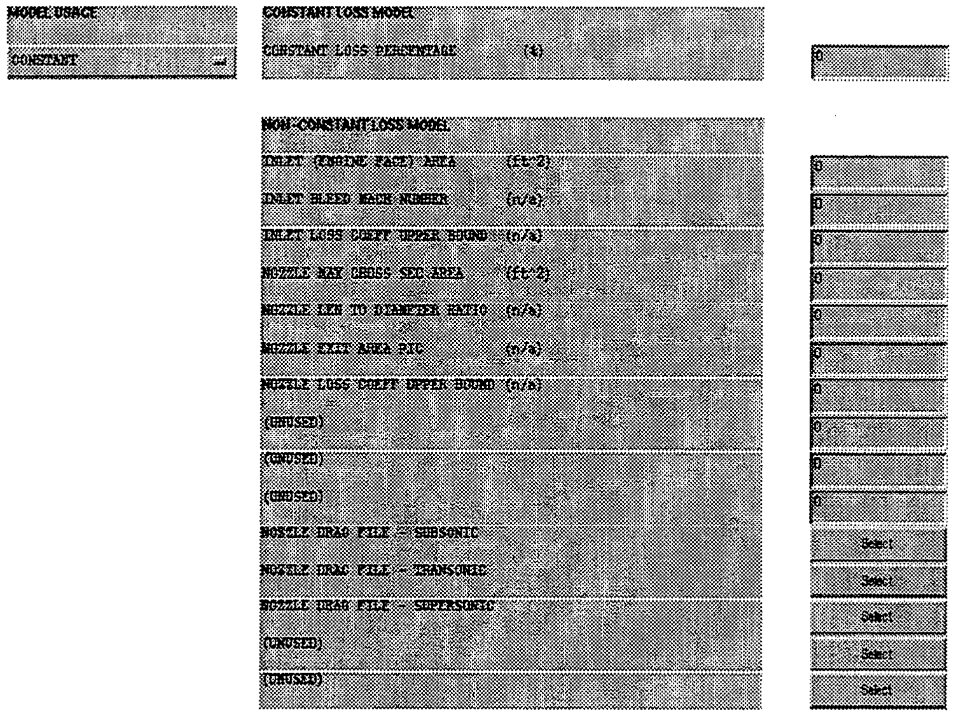
units, is given in Table 1.4.

Table 1.4: Mission Constants

Name	*Units*	*Description*
Thrust Percentage Increase	%	This parameter defines, as a percentage of installed thrust, the amount to increase installed thrust prior to using it for mission leg calculations. This is necessary in order to avoid potential errors which are not warranted. For example, suppose 6,000 pounds of installed thrust are required for a thrust-throttling leg. Without this percentage increase, even an installed thrust value of 5999.9 would be interpreted as a failure.
Initial Weight Fraction	n/a	This parameter defines, as a fraction of aircraft gross takeoff weight, the aircraft weight at the beginning of a mission. Any value between 0 and 1 may be used.
Leg Failure Option	n/a	This parameter affects the manner in which mission evaluation is performed. It is used whenever a given leg fails for any reason, such as insufficient thrust. A value of 1 allows a leg failure, after which mission evaluation continues as if the failed leg was not contained in the profile. *The mission output is flagged to indicate this occurrance, and all other output data corresponding to the failed leg should be considered void.* A value of 2 does not allow a leg failure, and if a given leg fails then mission evaluation will terminate. Only values of 1 and 2 may be used.
A/B Throttle Percentage Tolerance	%	This parameter defines, as a percentage of uninstalled thrust, the difference between required uninstalled thrust and the amount of uninstalled thrust provided by TERMAP which suffices for convergence during afterburner throttle iteration.
A/B Throttle Maximum Iterations	n/a	This parameter defines the maximum number of iterations which are permitted during afterburner throttling.

1.2.14 Installation Loss Model

This data element contains the variables necessary for determining the effect of installation losses during mission evaluation. The format of this data element is shown in Figure 1.13.



MODEL USAGE

CONSTANT

CONSTANT LOSS MODEL	
CONSTANT LOSS PERCENTAGE (%)	0
NON-CONSTANT LOSS MODEL	
INLET (ENGINE FACE) AREA (ft^2)	0
INLET BLEED MACH NUMBER (n/a)	0
INLET LOSS COEFF UPPER BOUND (n/a)	0
NOZZLE MAX CROSS SEC AREA (ft^2)	0
NOZZLE LEN TO DIAMETER RATIO (n/a)	0
NOZZLE EXIT AREA PIC (n/a)	0
NOZZLE LOSS COEFF UPPER BOUND (n/a)	0
(UNUSED)	0
(UNUSED)	0
(UNUSED)	0
NOZZLE DRAG FILE - SUBSONIC	Select
NOZZLE DRAG FILE - TRANSONIC	Select
NOZZLE DRAG FILE - SUPERSONIC	Select
(UNUSED)	Select
(UNUSED)	Select

Figure 1.13: Installation Loss Model

Model Usage

The selected type defines the installation loss model which will be used. Both a constant and a non-constant loss model are supported.

Constant Loss Model

For a constant loss model, a scalar loss percentage is the only data required.

Non-Constant Loss Model

For a non-constant loss model, a variety of scalar quantities and files are required from the user. A description of each of these is given in Table 1.5.

Table 1.5: Non-Constant Loss Model Data

Name	*Units*	*Description*
Inlet (Engine Face) Area	sq ft	This parameter defines the size of the physical engine inlet.
Inlet Bleed Mach Number	n/a	This parameter defines the mach number at which bypass and bleed flows leave the inlet for supersonic cases.

Non-Constant Loss Model Data - continued

Name	*Units*	*Description*
Inlet Loss Coeff Upper Bound	n/a	This parameter defines a realistic upper bound for the inlet loss coefficient, where the coefficient is interpreted as inlet drag force divided by uninstalled thrust.
Nozzle Max Cross Sec Area	sq ft	This parameter defines the size of the largest cross sectional area portion of the nozzle.
Nozzle Len to Diameter Ratio	n/a	This parameter defines the length to diameter ratio of the nozzle, where the diameter at the largest cross sectional area is used.
Nozzle Exit Area PIC	n/a	This parameter defines the (cycle-dependent) TERMAP PIC which provides the size of the nozzle exit. *It is assumed that TERMAP gives nozzle exit area in square inches for all cycles.*
Nozzle Loss Coeff Upper Bound	n/a	This parameter defines a realistic upper bound for the nozzle loss coefficient, where the coefficient is interpreted as nozzle drag force divided by uninstalled thrust.
Nozzle Drag File - Subsonic	n/a	An overview on the use and limitations of a subsonic nozzle drag file is given in Section 1.3.3.
Nozzle Drag File - Transonic	n/a	An overview on the use and limitations of a transonic nozzle drag file is given in Section 1.3.4.
Nozzle Drag File - Supersonic	n/a	An overview on the use and limitations of a supersonic nozzle drag file is given in Section 1.3.5.

1.2.15 Design Variable Configuration

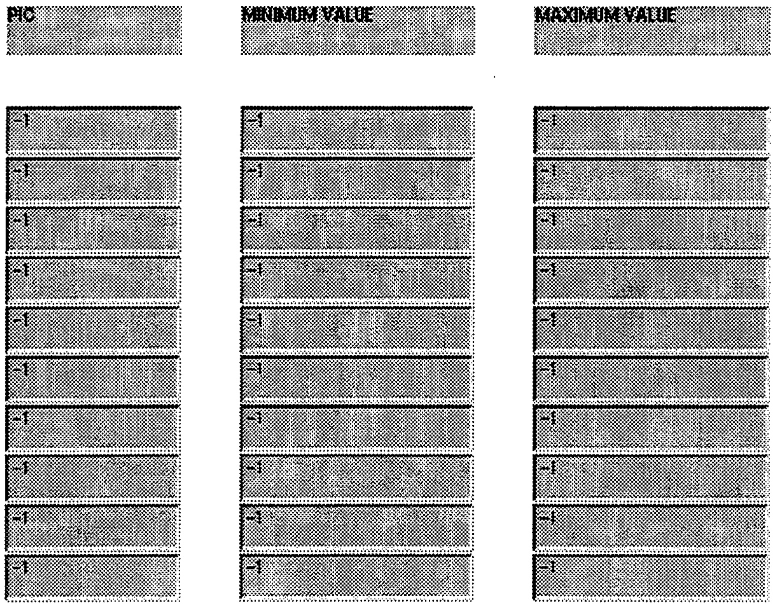
This data element contains a set of PICs, with ranges for each, which will be used as design variables for an optimization routine. The format of this data element is shown in Figure 1.14.

Unused entries must have a PIC value set to -1. Used entries must have an integer PIC value in the range from 1 to 2288.

1.2.16 Design Objectives with Aircraft Mission

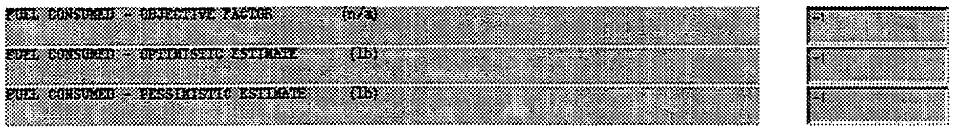
This data element contains an objective configuration for optimization cases involving an aircraft mission. The format of this data element is shown in Figure 1.15.

At present, the amount of fuel consumed during the aircraft mission is the only objective supported. The constants in this data element serve the purpose of scaling the fuel consumption objective and, as a mono objective case, the scaling will not significantly impact optimizer performance. A value of 1 should be used for the objective factor.



PIC	MINIMUM VALUE	MAXIMUM VALUE
-1	-1	-1
-1	-1	-1
-1	-1	-1
-1	-1	-1
-1	-1	-1
-1	-1	-1
-1	-1	-1
-1	-1	-1
-1	-1	-1
-1	-1	-1

Figure 1.14: Design Variable Configuration



FUEL CONSUMED - OBJECTIVE FACTOR (n/a)	-1
FUEL CONSUMED - OPTIMISTIC ESTIMATE (lb)	-1
FUEL CONSUMED - PESSIMISTIC ESTIMATE (lb)	-1

Figure 1.15: Design Objectives with Aircraft Mission

1.2.17 Design Objectives without Aircraft Mission

This data element contains an objective configuration for optimization cases not involving an aircraft mission. The format of this data element is shown in Figure 1.16.

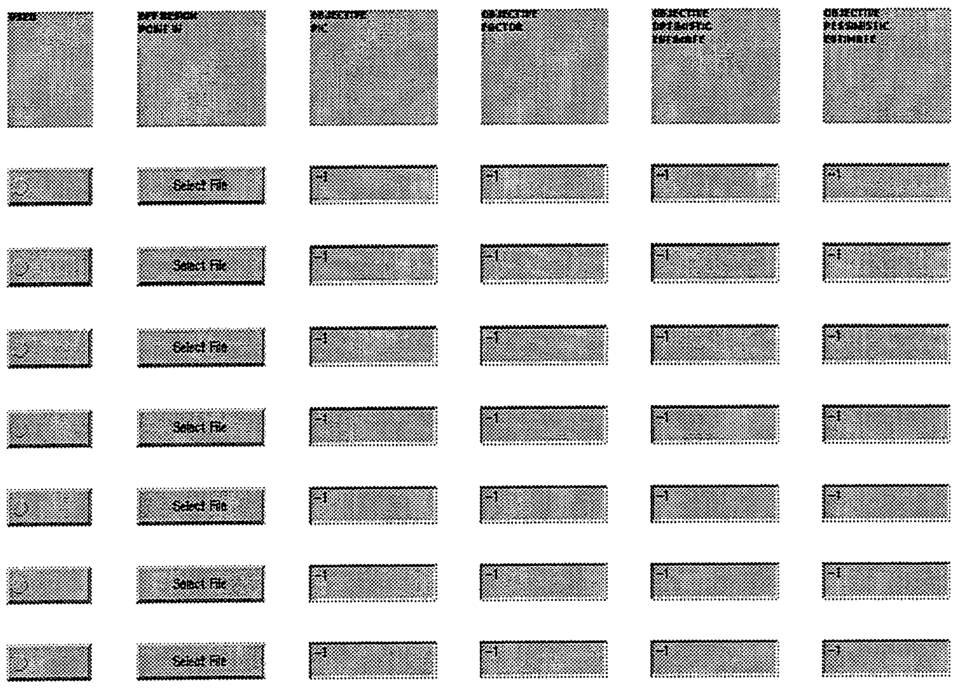


Figure 1.16: Design Objectives without Aircraft Mission

Off Design Points

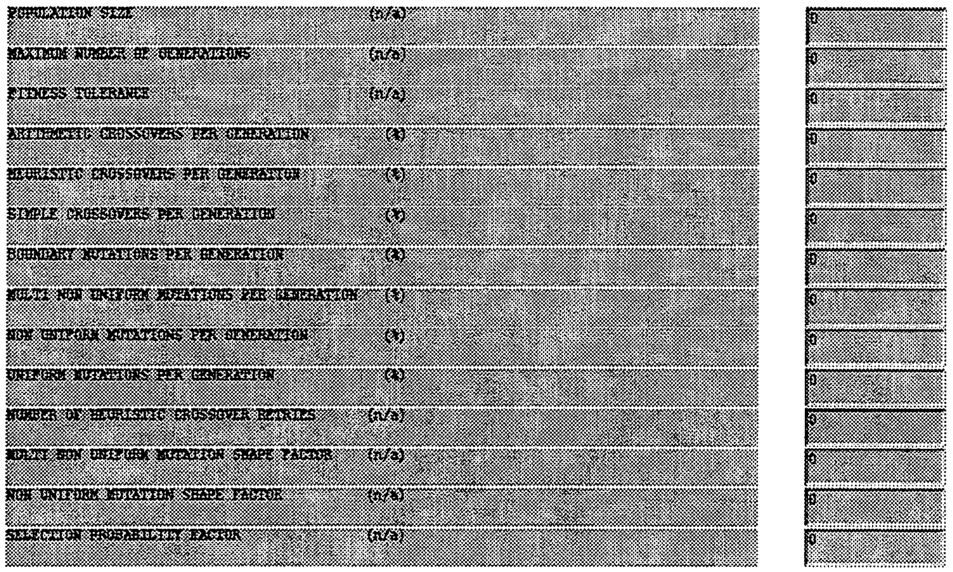
It is required that each off design point specified by this data element be contained in a Matlab data file. Upon selection of the *Select File* button, a standard file selection dialog will be provided in which the user can select an individual file containing the off design point.

Objective Scaling

The value of each individual objective is scaled to ensure that each objective will be of the same order of magnitude. The optimistic and pessimistic estimates serve this purpose, and the program uses these to scale each objective in the range from 0 to 1. The objective factors are a measure of importance for each individual objective, and each should be a positive number. By implementing objective factors such that the sum of all used objective factors equals 1, then the overall multi-objective value for a given design will be in the approximate range from 0 to 1.

1.2.18 Genetic Algorithm Options

This data element contains all definable parameters for the genetic algorithm optimizer. The format of this data element is shown in Figure 1.17. A description of each of these constants is



POPULATION SIZE	(n/a)	0
MAXIMUM NUMBER OF GENERATIONS	(n/a)	0
FITNESS TOLERANCE	(n/a)	0
ARITHMETIC CROSSOVERS PER GENERATION	(%)	0
HEURISTIC CROSSOVERS PER GENERATION	(%)	0
SIMPLE CROSSOVERS PER GENERATION	(%)	0
BOUNDARY MUTATIONS PER GENERATION	(%)	0
MULTI NON UNIFORM MUTATIONS PER GENERATION	(%)	0
NON UNIFORM MUTATIONS PER GENERATION	(%)	0
UNIFORM MUTATIONS PER GENERATION	(%)	0
NUMBER OF HEURISTIC CROSSOVER RETRIES	(n/a)	0
MULTI NON UNIFORM MUTATION SHAPE FACTOR	(n/a)	0
NON UNIFORM MUTATION SHAPE FACTOR	(n/a)	0
SELECTION PROBABILITY FACTOR	(n/a)	0

Figure 1.17: Genetic Algorithm Options

provided throughout the remainder of this section.

Population Size (n/a) This parameter defines the number of individual designs which will be used to initiate the simulated evolution. Until additional testing is performed, a value of 100 is recommended.

Maximum Number of Generations (n/a) This parameter defines the maximum number of times an offspring population will be created from a parent population. Until additional testing is performed, a value of 10 is recommended.

Fitness Tolerance (n/a) This parameter defines, with respect to the objective function value, the distance required for two designs to differ. This number should be small compared to the objective value, and a value of 1E-6 is recommended.

Arithmetic Crossovers per Generation (%) This parameter defines, as a percentage of population size, the number of times a crossover of the arithmetic type is applied per evolution cycle. Until additional testing is performed, a value of 5 is recommended.

Heuristic Crossovers per Generation (%) This parameter defines, as a percentage of population size, the number of times a crossover of the heuristic type is applied per evolution cycle. Until additional testing is performed, a value of 5 is recommended.

Simple Crossovers per Generation (%) This parameter defines, as a percentage of population size, the number of times a crossover of the simple type is applied per evolution cycle. Until additional testing is performed, a value of 5 is recommended.

Boundary Mutations per Generation (%) This parameter defines, as a percentage of population size, the number of times a mutation of the boundary type is applied per evolution cycle. Until additional testing is performed, a value of 5 is recommended.

Multi-Non-Uniform Mutations per Generation (%) This parameter defines, as a percentage of population size, the number of times a mutation of the multi non uniform type is applied per evolution cycle. Until additional testing is performed, a value of 5 is recommended.

Non-Uniform Mutations per Generation (%) This parameter defines, as a percentage of population size, the number of times a mutation of the non uniform type is applied per evolution cycle. Until additional testing is performed, a value of 5 is recommended.

Uniform Mutations per Generation (%) This parameter defines, as a percentage of population size, the number of times a mutation of the uniform type is applied per evolution cycle. Until additional testing is performed, a value of 5 is recommended.

Number of Heuristic Crossover Retries (n/a) This parameter defines the total number of times attempts can be made to create a valid child design from two parent designs by applying a crossover of the heuristic type. Until additional testing is performed, a value of 3 is recommended.

Multi-Non-Uniform Mutation Shape Factor (n/a) This parameter affects the amount of change a design variable can undergo during a mutation of the multi non uniform type. A value greater than or equal to the maximum number of generations should always be used.

Non-Uniform Mutation Shape Factor (n/a) This parameter affects the amount of change a design variable can undergo during a mutation of the non uniform type. A value greater than or equal to the maximum number of generations should always be used.

Selection Probability Factor (n/a) This parameter affects the manner in which designs are carried over to a new population. Until additional testing is performed, a value of 0.08 is recommended.

1.2.19 Kriging Options

This data element defines whether or not kriging is to be used during optimization routines. Its format is a simple pulldown menu which only contains only two options: *Use Kriging* or *Do Not Use Kriging.*

1.2.20 Output - Genetic Algorithm End Population

This data element contains ASCII text, in matrix format, which represents the population of designs existing at the end of the simulated evolution. Each row corresponds to an individual design, with the values for each design variable and the overall objective value given. Column headers are provided within the text for interpretation.

1.2.21 Output - Genetic Algorithm Best Population

This data element contains ASCII text, in matrix format, which represents the overall best design in each generation during the simulated evolution. Each row corresponds to a generation, with the generation number, values for each design variable, and the overall objective value given. Column headers are provided within the text for interpretation.

1.2.22 Output - Genetic Algorithm Trace Information

This data element contains ASCII text, in matrix format, which represents the best and average objective values for all designs within each generation. Each row corresponds to a generation, with the generation number, best objective value, and average objective value given. Column headers are provided within the text for interpretation.

1.2.23 Output - Best Solution

This data element contains ASCII text which represents the overall best design found during the simulated evolution. The values for each design variable and the overall objective value are given. Identifiers are provided within the text for interpretation.

1.2.24 Output - Cycle Analysis

This data element contains ASCII text, divided into an on-design section and an off-design section, which provides cycle performance data. The cycle data is organized by parameter index code, and identifiers within the text permit interpretation.

1.2.25 Output - Mission Analysis

This data element contains ASCII text, divided by flight leg, which provides mission performance data. Each mission leg has data fields as given in Table 1.6. Fields beginning with => represent data values for a single engine.

Table 1.6: Mission Analysis Output

Name	*Units*	*Description*
Leg Number	n/a	self explanatory
Leg Type	n/a	self explanatory
Error Code	n/a	Discussion of this component is given in Section 1.6.2.
Warning Code	n/a	Discussion of this component is given in Section 1.6.2.
Initial Velocity	ft/s	self explanatory
Representative Velocity	ft/s	self explaatory
Final Velocity	ft/s	self explanatory
Initial Altitude	ft	self explanatory
Representative Altitude	ft	self explanatory
Final Altitude	ft	self explanatory
Initial Mach	n/a	self explanatory
Representative Mach	n/a	self explanatory
Final Mach	n/a	self explanatory

Mission Analysis Output - continued

Name	*Units*	*Description*
Dynamic Pressure	psf	self explanatory
Lift Coefficient	n/a	self explanatory
Drag Coefficient	n/a	self explanatory
Drag Force	lb	self explanatory
=> IFAIL	n/a	The TERMAP `IFAIL` parameter.
=> MODE	n/a	The TERMAP `MODE` parameter.
=> PLA	n/a	The TERMAP `PLA` parameter.
=> PCTRH(1)	n/a	The TERMAP `PCTRH(1)` parameter.
=> SFC	1/s	self explanatory
=> TSFC	1/s	self explanatory
=> SFC	1/hr	self explanatory
=> TSFC	1/hr	self explanatory
=> Uninstalled Thrust	lb	The uninstalled thrust provided by TERMAP.
=> Inlet Loss Coefficient	n/a	self explanatory
=> Nozzle Loss Coefficient	n/a	self explanatory
=> Installed Thrust	lb	self explanatory
=> Installed Thrust W/Tol	lb	The installed thrust with the percentage increase defined by the user.
Total Installed Thrust	lb	The total amount of installed thrust value which the A/C has available. This will only be different from the previous variable if there are multiple engines.
Excess Thrust	lb	The difference between drag force and total installed thrust.
Climb Angle	rad	self explanatory
Climb Angle	deg	self explanatory
Vertical Distance	ft	self explanatory
Horizontal Distance	ft	self explanatory
Total Distance	ft	self explanatory
Vertical Distance	nm	self explanatory
Horizontal Distance	nm	self explanatory
Total Distance	nm	self explanatory
Vertical Velocity	ft/s	self explanatory
Horizontal Velocity	ft/s	self explanatory
Total Velocity	ft/s	self explanatory
Time	sec	self explanatory
Time	min	self explanatory
Time	hr	self explanatory
Initial Weight	lb	self explanatory
Final Weight	lb	self explanatory
Weight Difference	lb	self explanatory

Mission Analysis Output - continued

Name	*Units*	*Description*
Initial Weight Fraction	n/a	self explanatory
Final Weight Fraction	n/a	self explanatory

Any data whose value is -1 signifies that it was not calculated (velocities for warmup legs, for example).

1.3 File Formats and Limitations

1.3.1 TERMAP Input File

A default TERMAP input file is required from the user. The format of this file is the same as that normally used for TERMAP operation with the following exceptions:

1. There must not be any `MAXLIM` or `ENGMAX` definitions within the file.
2. There must not be any `MINLIM` or `ENGMIN` definitions within the file.
3. There must be one, and only one, off design point line defined in the file. Furthermore, this line must be `$D IDES=0, LAST=1, $D`.

Further information about the use of a baseline TERMAP input file with the program may be found in the SDD.

1.3.2 Aircraft Drag File

The following is taken verbatim from *Aircraft Engine Design* by Mattingly, et al.

The conventional form of the lift-drag polar equation is

$$C_D = C_{D\min} + K'C_L^2 + K''(C_L - C_{L\min})^2$$

where K' is the inviscid drag due to lift (induced drag) and K'' is the viscous drag due to lift (skin friction and pressure drag). Expanding and collecting like terms shows that the lift-drag polar equation may also be written

$$C_D = (K' + K'')C_L^2 - (2K''C_{L\min})C_L + (C_{D\min} + K''C_{L\min}^2)$$

or

$$C_D = K_1C_L^2 + K_2C_L + C_{Do} \tag{1.1}$$

where

$$K_1 = K' + K''$$

$$K_2 = -2K''C_{L\min}^2$$

$$C_{Do} = C_{D\min} + K''C_{L\min}^2$$

Note that the physical interpretation of $C_{Do}$is the drag coefficient at zero lift. Also, for most high performance aircraft $C_{L\min} \approx 0$, so that $K_2 \approx 0$.

Equation 1.1 is the expression used to calculate drag coefficients within the program. The values for K_1, K_2, and C_{Do} at any flight mach number are found by linear interpolation of the drag table specified within the drag file. The drag file must *only* contain numbers in a matrix format, no comments are permitted. Column 1 corresponds to mach number, column 2 corresponds to K_1, column 3 corresponds to K_2, and column 4 corresponds to C_{Do}. Since the values are linearly interpolated, rather than approximated from curve fitting, the resolution of the drag profile has an impact. Also, extrapolation of the data is not supported, so data for a zero mach number as well as a sufficiently high mach number should be provided.

1.3.3 Subsonic Nozzle Drag File

For the cases where the flight mach number is less than 0.8, and a non-constant installation loss model is used, a subsonic nozzle drag file is required. This file must *only* contain numbers in a matrix format, no comments are permitted. Column 1 corresponds to IMS, and column 2 corresponds to C_D, where IMS and C_D are defined in *Aircraft Engine Design* by Mattingly, et al.

1.3.4 Transonic Nozzle Drag File

For the cases where flight mach number is between 0.8 and 1.2, and a non-constant installation loss model is used, a transonic nozzle drag file is required. This file must *only* contain numbers in a matrix format, no comments are permitted. Column 1 corresponds to M_0, and column 2 corresponds to C_{DP}, where M_0 and C_{DP} are defined in *Aircraft Engine Design* by Mattingly, et al.

1.3.5 Supersonic Nozzle Drag File

For the cases where flight mach number is greater than 1.2, and a non-constant installation loss model is used, a supersonic nozzle drag file is required. This file must *only* contain numbers in a matrix format, no comments are permitted. Column 1 corresponds to IMS, and column 2 corresponds to $C_D|_{M_0=1.2}$, where IMS and $C_D|_{M_0=1.2}$ are defined in *Aircraft Engine Design* by Mattingly, et al.

1.3.6 Matlab Data Files

When saving and loading data elements to and from a disk file, the program uses Matlab's (*.mat) format. This is a type of machine language, and is accessible only through Matlab.

1.4 First Time Setup

The DoD-Limited CD-ROM separate from this document contains the following:

- The directory `Data` contains example ASCII text data files.
- The directory `Developer` contains a variety of files which may be useful for further modification of the overall package.
- The directory `Documentation` contains all documentation, in LaTeX format, for the program.
- The directory `Fortran` contains the original and modified versions of TERMAP, both FORTRAN-77 code and executables, which were used during development on a UNIX platform.
- The directory `Functional_Areas` contains Matlab source code.
- The directory `GA` contains Matlab source code.
- The directory `GUI` contains Matlab source code.
- The directory `Initialization` contains Matlab source code.
- The directory `Inst_Loss` contains Matlab source code.
- The directory `Kriging` contains Matlab source code.
- The directory `Lower_Level` contains Matlab source code.
- The directory `Mission` contains Matlab source code.
- The directory `TERMAP_IO` contains Matlab source code.
- The file `Script.m` is a single unit of Matlab code.

With the exception of the `Developer`, `Documentation`, and `Fortran` directories, each of the above mentioned items should be copied to the same parent directory on the user's machine. At this point, there are two additional steps required. First, the user must place a TERMAP executable (appropriately modified for use with the program) in the `TERMAP_IO` directory. Second, the user must make changes to the file `Script.m` in order to permit Matlab to access all source code and executable files comprising the program. Comments within `Script.m` provide detailed instruction as to the specific changes required.

It should also be noted that all font information for the screen displays is controlled by the file `Init_Graphics.m` within the `GUI` directory. Depending on the machine and monitor being used, there may be changes required to this file as well. Comments within `Init_Graphics.m` provide detailed instruction as to the specific changes required.

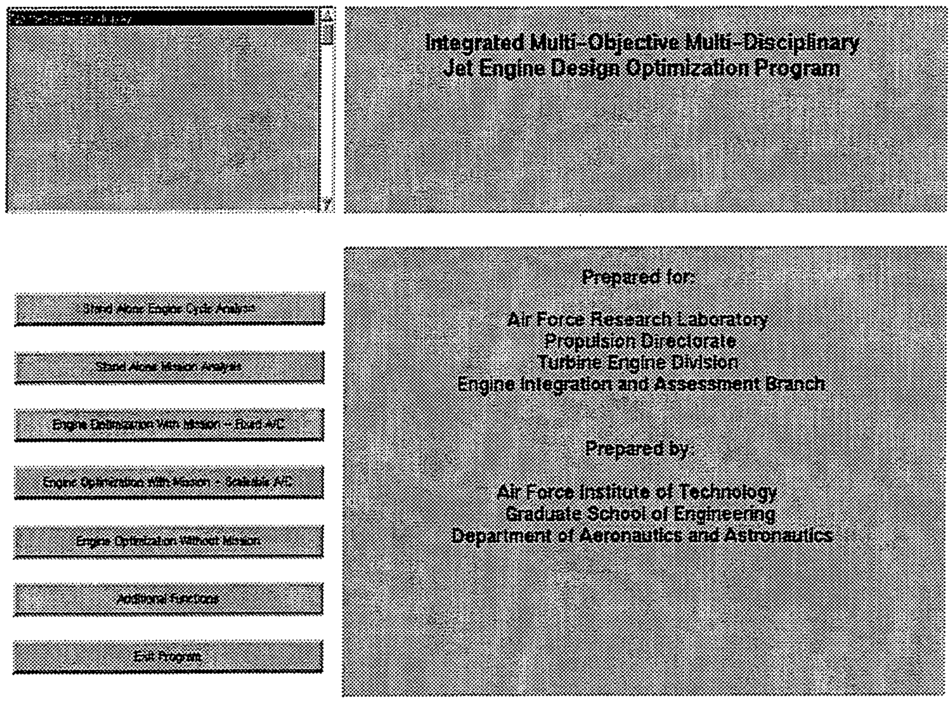
1.5 Running the Program

Getting the program up and running is a three stage process.

1. Initiate Matlab.
2. At the Matlab command line prompt, change the working directory to the one which contains the file `Script.m`.
3. At the Matlab command line prompt, type `Script` and hit return.

1.5.1 Primary Menu

The primary menu is the first to be displayed on the screen, and serves as a base menu from which all areas of the program may be accessed. It is shown in Figure 1.18.


Stand Alone Engine Cycle Analysis
Stand Alone Mission Analysis
Engine Optimization With Mission - Fixed A/C
Engine Optimization With Mission - Scalable A/C
Engine Optimization Without Mission
Additional Functions
Exit Program

Integrated Multi-Objective Multi-Disciplinary
Jet Engine Design Optimization Program

Prepared for:

Air Force Research Laboratory
Propulsion Directorate
Turbine Engine Division
Engine Integration and Assessment Branch

Prepared by:

Air Force Institute of Technology
Graduate School of Engineering
Department of Aeronautics and Astronautics

Figure 1.18: Primary Menu

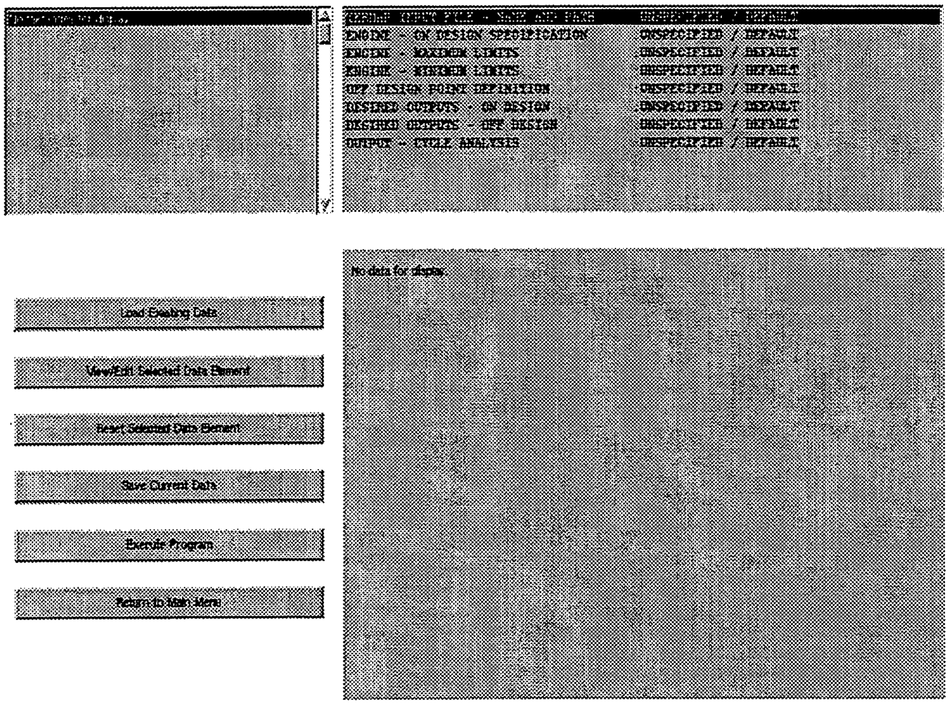
The available options for this menu are:

- *Stand Alone Engine Cycle Analysis* This option transfers the user to the corresponding menu for this functional area.
- *Stand Alone Mission Analysis* This option transfers the user to the corresponding menu for this functional area.
- *Engine Optimization With Mission - Fixed A/C* This option transfers the user to the corresponding menu for this functional area.

- *Engine Optimization With Mission - Scaleable A/C* This option transfers the user to the corresponding menu for this functional area.
- *Engine Optimization Without Mission* This option transfers the user to the corresponding menu for this functional area.
- *Additional Functions* This option transfers the user to the Additional Functions Menu.
- *Exit Program* This option, after user confirmation, terminates the program as well as Matlab.

1.5.2 Main Functional Area Menus

These menus are all identical in operation, and for illustrative purposes the Stand Alone Engine Cycle Analysis Menu will be used throughout this section. This menu is shown in Figure 1.19.


```
[illegible]                              [illegible]
ENGINE - ON DESIGN SPECIFICATION         UNSPECIFIED / DEFAULT
ENGINE - MAXIMUM LIMITS                  UNSPECIFIED / DEFAULT
ENGINE - MINIMUM LIMITS                  UNSPECIFIED / DEFAULT
OFF DESIGN POINT DEFINITION              UNSPECIFIED / DEFAULT
DESIRED OUTPUTS - ON DESIGN              UNSPECIFIED / DEFAULT
DESIRED OUTPUTS - OFF DESIGN             UNSPECIFIED / DEFAULT
OUTPUT - CYCLE ANALYSIS                  UNSPECIFIED / DEFAULT
```

No data for display.

Figure 1.19: Stand Alone Engine Cycle Analysis Menu

The available options for this menu are:

- *Load Existing Data* This option initiates the process by which the user may load data from file into the current menu data structure. Additional information concerning this option is covered later in this section.
- *View/Edit Selected Data Element* This option brings up the contents of the data element which is highlighted in the status area. Section 1.2 provides details concerning data elements.

- *Reset Selected Data Element* This option, after user confirmation, clears the data element contents to its original state.
- *Save Current Data* This option initiates the process by which the user may save data from the current data structure to file. Additional information concerning this option is covered later in this section.
- *Execute Program* This option initiates the process by which the core function for the menu is evaluated.
- *Return to Main Menu* This option returns the user to the Primary Menu.

Saving Data to File

Upon selecting the Save Current Data option, the user will have available the following options (which should be completed in the order given):

- *Select Data Elements* This option brings up a list of data elements corresponding to the displayed menu. The user then selects the desired data elements to save.
- *Assign Filename* This option brings up a standard file selection dialog for saving a data file. The *.mat extension must always be used.
- *Save To File* This option physically transfers the data from the current menu data structure to file.
- *Done - Cancel* This option completes or cancels the process.

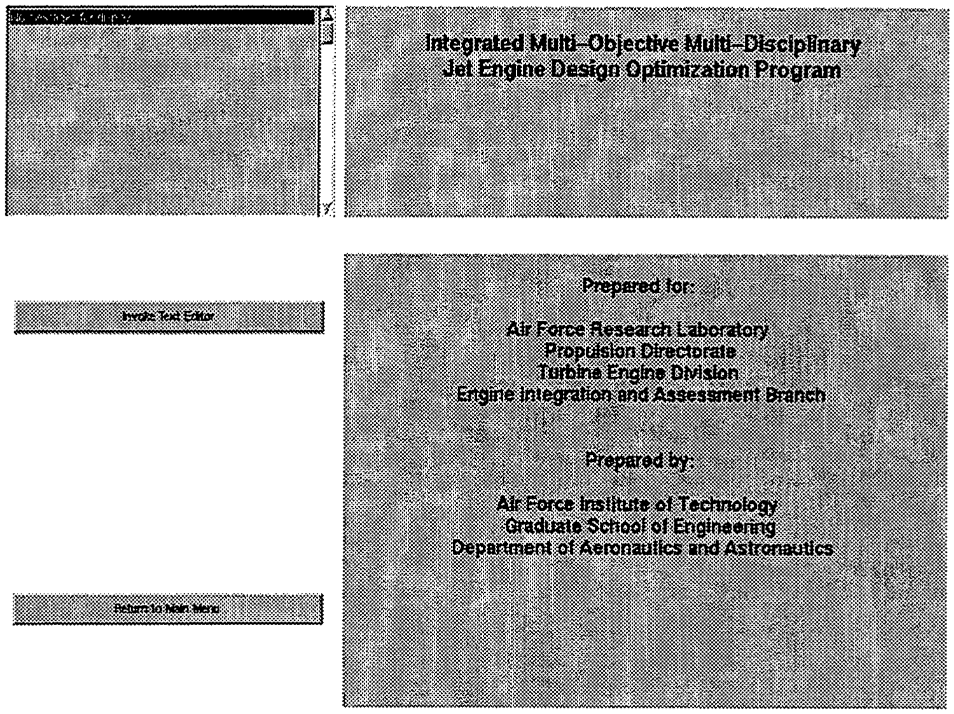
Loading Data from File

Upon selecting the Load Existing Data option, the user will have available the following options (which should be completed in the order given):

- *Select Data File* This option brings up a standard file selection dialog for choosing a data file. Only *.mat files may be used.
- *Select Available Data Elements* This option brings up a list of the data elements contained in the file which are also applicable to the displayed menu. The user then selects the desired data elements to load.
- *Load Data* This option physically transfers the data from file into the current menu data structure.
- *Done - Cancel* This option completes or cancels the process.

1.5.3 Additional Functions Menu

This menu serves as an area to perform secondary functions within the program, and is shown in Figure 1.20. At present, the only secondary function which may be performed from this menu is invoking a text editor.


Integrated Multi-Objective Multi-Disciplinary
Jet Engine Design Optimization Program

Prepared for:

Air Force Research Laboratory
Propulsion Directorate
Turbine Engine Division
Engine Integration and Assessment Branch

Prepared by:

Air Force Institute of Technology
Graduate School of Engineering
Department of Aeronautics and Astronautics

Figure 1.20: Additional Functions Menu

1.6 Limitations

This section describes current limitations of the program.

1.6.1 Genetic Algorithm Optimizer

It is possible for the end population to contain a design vector whose objective value is better than that described as the best overall solution. This is a limitation within the original GAOT package, and no correction has been implemented. The user should check the end population for this possible occurance.

The recommended values for the GA parameters as described in Section 1.2.18 are provided for initial testing purposes only. The performance of GA optimizer is case dependent, and experience gained by using the program will permit the user to customize these parameters according to specific classes of problems.

1.6.2 Mission Evaluation

It is possible, during a constant speed climb mission leg evaluation, for the total installed thrust value obtained in a non-thrust throttling mode to *exceed* that which corresponds to a 90 degree flight angle. This means that even if flight is vertical, the aircraft will still be accelerating. For these cases, a thrust throttling mode is used where the installed thrust yields a 45 degree flight angle.

The use of the Mattingly-based non-constant installation loss model requires some a priori knowledge of the required engine sizing parameters corresponding to a given mission profile. A poor selection of these sizing parameters could produce unrealistic values for the inlet and nozzle loss coefficients, and in some cases it may not even be possible to calculate them. Rigorous provisions have been introduced into the code which create and use an overall pass/fail status of the non-constant loss model. If the coefficients are out of bounds (i.e. negative or greater than the user defined upper limits), or if the coefficients can't be calculated, then this is interpreted as a loss model failure. A loss model failure ultimately translates to a mission leg failure.

The error and warning code fields in the mission analysis output are displayed as numbers, and correspond to the following:

0 No errors or warnings.

100 The constraint for the leg was not recognized. *This was created for development purposes and will never occur.*

200 Data table linear interpolation failure.

300 Lift coefficient violation.

400 Insufficient thrust.

500 The constraint for the leg, such as distance or time, was not satisfied.

600 The afterburner option for the leg was not recognized. *This was created for development purposes and will never occur.*

700 Takeoff distance constraint not satisfied.

800 Leg failure continuation.

900 Afterburner throttling failed.

1000 TERMAP physically crashed.

1100 TERMAP returned an infeasible point.

1200 Unexpected result.

1300 Unable to determine uninstalled engine settings which produce a required installed thrust.

1400 Installation loss model failure.

1.6.3 TERMAP

Under extremely rare circumstances, it is possible for TERMAP to produce a series of asterisks (*) instead of a numerical quantity for one or more output variables. All TERMAP I/O is built entirely around PIC codes and their corresponding data, and no error checking has been built in to search for non-numerical data within the TERMAP output. A complete program failure will result if this occurs. This type of behavior has only been observed during testing specifically designed to exercise TERMAP, and has not occured at any other time during development.

1.6.4 Graphical User Interface

If the user enters a non-numerical entry in a data field requiring a numerical entry, and then confirms his selection, a Matlab error will occur. Although the program will function properly afterward, it is recommended that the user restart the program.

When executing one of the main functions within the program, it is possible for the message area to become obscured (in some cases it may even be blank). This problem is due to Matlab's internal queing process, and cannot be corrected. If the user has any doubt as to whether or not the code has hung, he or she may check the Matlab command window for information which is continuously displayed throughout processing.

1.6.5 Kriging

The required interface between all functional areas which perform optimization and the kriging code is complete. For trivial cases with only one design variable, the use of kriging has been successful. For more meaningful cases with two or more design variables, however, attempts at using kriging have failed (the kriging code physically crashes). Although the interface specification described in *Improving Algorithic Efficiency of Aircraft Engine Design for Optimal Performance* by Millhouse has been followed, the most likely cause of the problem is due to a high-level interfacing issue. All of the lower level algorithms within the routine were previously validated for cases involving many design variables, and were shown to significantly enhance the overall engine optimization process. With some additional effort, this problem can be remedied.

Chapter 2

Software Design Document

2.1 Overview

This purpose of this Software Design Document (SDD) to describe the overall design of the Integrated Multi-Objective Multi-Disciplinary Jet Engine Design Optimization Program. Details have been included to describe the decomposition of the package into modules and the high-level data structures comprising them.

2.1.1 Scope

There are three important aspects that, with respect to the overall scope of this SDD, need to be addressed. First, there are no formal software requirements corresponding to development. Regular meetings between the developer and the funding organization have served as a means by which to agree on functionality and check progress. Second, an explicit description of calculations performed within the program is not given. All calculations are consistent with the referenced documents. Third, this SDD provides a high-level description of the overall design. The developer has spent considerable effort on making the source code as readable and self-documented as possible, and lower-level details may be found in the source code itself.

2.1.2 Intended Reader

This SDD is written for individuals who are familiar with the Matlab programming language and the Turbine Engine Reverse Modeling Aid Program (TERMAP). Individuals not well versed in these areas, however, should have have little difficulty in navigating through this document and understanding the basic concepts.

2.1.3 Pertinent Matlab Concepts

With the exception of TERMAP, all of the program's source code is written in the Matlab programming language. Table 2.1 gives a listing of Matlab data types referenced throughout this SDD.

Table 2.1: Matlab Data Types

Reference Name	*Description*
scalar	This is a scalar quantity. Although Matlab supports complex variables, none are used anywhere within the program.
vector	This is a vector of data. Vectors can be of either row or column types. A vector with only one entry is equivalent to a scalar.
matrix	This is a matrix of data. A matrix with only one row or one column is equivalent to a vector.
string	This is a character string.

Matlab Data Types - continued

Reference Name	*Description*
cell	This is a cell array, one of Matlab's most powerful programming features. It is similar to a matrix, except that index entries can be *any* data type, including other cell arrays. The `cell` Matlab command may be used to create it.
struct	This is a data structure, another powerful programming feature. It is very similar to structures in the C/C++ languages. The `struct` Matlab command may be used to create it.
cell-struct	This reference name signifies a combination of the cell and struct data types. This is not a standard Matlab data type.
uicontrol	This is a user interface control, Matlab's way of manipulating GUI objects. The `uicontrol` Matlab is required to create it.
figure	This is a figure, and is another Matlab user interface construct. The `figure` Matlab command is required to create it.

Another important Matlab concept is that of global data. By defining any specific variable (regardless of data type) to be global, it is accessible from any Matlab unit of code. This is analagous to common data blocks in the FORTRAN-77 language.

Lastly, it is important to note that each unit of Matlab code is used as either a *function* or as a *procedure*. This distinction is critical in terms of accessing and modifying data, and is analogous to functions and procedures in the C/C++ languages.

2.1.4 Pertinent TERMAP Concepts

There are two important aspects of TERMAP that have shaped certain design characteristics of this package.

First is the concept of a parameter index code (PIC). Each data variable within TERMAP has its own unique PIC, and through the use of FORTRAN-77 equivalence statements each variable is accessible and/or modifiable through this identification number. This is extremely convienent from a programming perspective, and all TERMAP I/O within this package has been built exclusively around PICs.

Second is a limitation with TERMAP itself. Specifically, TERMAP cannot be used in a thrust-throttling mode in order to a attain a desired thrust with the afterburner engaged. The workaround employed in this package is to iteratively use TERMAP in a non-thrust throttling mode while varying the afterburner setting. A Newton-Raphson scheme has been employed for these iterations, and works quite well. For accuracies within 1%, a total of 2 iterations are usually required. For accuracies within 0.1%, an order of magnitude increase, a total of 4 iterations are usually required.

2.2 Overall Structure and Decomposition

All files comprising the package are contained in a number of directories. Each of these directories represents a logical grouping of related files, and the term module is used to represent this association. The modules comprising the program are outlined below.

Data This module contains user supplied/generated data files.

Genetic Algorithm This module contains Matlab code to perform function optimization using genetic algorithms.

Graphical User Interface This module contains Matlab code to control the graphical user interface.

Initialization This module contains Matlab code to initialize global data used throughout the program.

Installation Loss This module contains Matlab code to calculate the effects of engine installation losses.

Functional Areas This module contains Matlab code to perform the core functions of the program.

Lower Level This module contains Matlab code to perform various utility-type tasks.

Mission This module contains Matlab code to perform mission analysis.

TERMAP I/O This module contains Matlab code to interface with TERMAP, along with a user-supplied executable version of TERMAP itself.

Kriging This module contains Matlab code to perform function estimation using kriging.

The decomposition of each module is given in individual sections throughout the remainder of this document. Again, these sections are intended to provide a higher level design description only.

2.3 Data Module

The data module contains files which are either supplied by the user or generated during program execution. In general, the program only supports files in ASCII text format and a Matlab-specific data storage format.

2.3.1 User Supplied ASCII Text Data Files

For the case where an ASCII text file is assumed to contain numbers in a matrix format, the `load` Matlab command is used to transfer the file contents to a matrix within Matlab memory.

The only other case where a user supplied ASCII text file is used by the program is for the manipulation of a baseline TERMAP input file. This file is *not*, however, loaded into Matlab memory. Further details encompassing the usage of a TERMAP input file by the program is covered in Section 2.12.

2.3.2 I/O With Matlab Data Storage Format

The program has been designed to interact with Matlab data files which contain two variables: `GLOBAL_ELEMENTS` and `GLOBAL_DATA`. The `GLOBAL_ELEMENTS` variable is a vector which defines what data elements are contained in the file. The `GLOBAL_DATA` variable is a combined cell array and data structure which physically contains data element contents. Refer to the source code for further details.

2.4 Genetic Algorithm Module

The genetic algorithm routines used in the program are based an existing public-domain package, named GAOT, developed by North Carolina State University. The fundamental operation of the adapted version is the same as the original GAOT package, although significant changes have been made to improve code efficiency and readability.

2.4.1 Purpose and High Level Operation

The purpose of the genetic algorithm module is to search for a design which optimizes a given objective function. The GA operates by simulating the process of natural selection, with individual designs treated as gene structures. Beginning with a randomly generated population of designs, a new generation is created by random variations (mutations) and matings (crossovers) of existing designs. This new generation is treated as a baseline population for another set of mutations and crossovers, and the process repeats itself until termination criteria is met.

A key concept inherent to genetic algorithms is the fitness of a design. Fitness is a measure of goodness, and must be a scalar quantity regardless of the number of design variables or the number of design objectives. The convention used with this GA is that the higher the fitness value the better. Maximization and minimization problems (or a combination of both for the case of multiple objectives) are possible with care given toward the scaling of objective quantities.

2.4.2 Accessing

The unit `Genetic_Algorithm` is the subprogram driver, and is accessed as a Matlab function. All inputs and outputs are passed in and out as data structures.

Inputs

There is a single input to the unit `Genetic_Algorithm`. It is a data structure with fields given in Table 2.2.

Table 2.2: Genetic Algorithm Input Fields

Field Name	*Data Type*	*Brief Description*
`DES_VAR_BOUNDS`	matrix	Ranges for design variables.
`OBJ_FCN`	string	Name of the objective function.
`POP_SIZE`	scalar	Population size.
`MAX_GEN`	scalar	Maximum number of generations.

Genetic Algorithm Input Fields - continued

Field Name	*Data Type*	*Brief Description*
`FIT_TOL`	scalar	Tolerance for two fitnesses to differ.
`AC_PER_GEN`	scalar	Number of arithmetic crossovers per generation.
`HC_PER_GEN`	scalar	Number of heuristic crossovers per generation.
`SC_PER_GEN`	scalar	Number of simple crossovers per generation.
`BM_PER_GEN`	scalar	Number of boundary mutations per generation.
`MNM_PER_GEN`	scalar	Number of multi non uniform mutations per generation.
`NM_PER_GEN`	scalar	Number of non uniform mutations per generation.
`UM_PER_GEN`	scalar	Number of uniform mutations per generation.
`HC_RETRIES`	scalar	Number of retries for a heuristic crossover.
`MNM_SHAPE`	scalar	Shape factor for a multi non-uniform mutation.
`NM_SHAPE`	scalar	Shape factor for non-uniform mutation.
`NGS_SEL_PROB`	scalar	Selection probability for designs to be selected to a new generation.

Outputs

There is a single output to the unit `Genetic_Algorithm`. It is a data structure with fields given in Table 2.3

Table 2.3: Genetic Algorithm Output Fields

Field Name	*Data Type*	*Brief Description*
`START_POP`	matrix	Starting population.
`END_POP`	matrix	Ending population.
`BEST_POP`	matrix	Best design vector for each generation.
`TRACE_INFO`	matrix	Best and average fitness values for each generation.
`OPT_DES_VEC`	vector	Overall best design vector found by the optimizer.

2.4.3 Global Data

Pre-Defined

There is no pre-defined global data required for operation.

Created for Processing

All global data created for processing by the GA begins with the prefix `GA_`. Each of the inputs described in Section 2.4.2 has an associated global data parameter. For example, the `DES_VAR_BOUNDS` data structure input has global data `GA_DES_VAR_BOUNDS`.

There are two additional global parameters defined. One is `GA_NUM_DES_VAR`, which is the number of design variables. The other is `GA_CURRENT_GEN`, and represents the instantaneous generation number during the simulated evolution.

2.4.4 Module Decomposition

A complete listing of all units within the GA module is given in Section 3.1. These routines are a modification of the existing GAOT package, and no further design infomation is given in this document.

2.5 Graphical User Interface Module

The graphical user interface routines used in the program have been created specifically for this project. The concept of data elements, as described in the SUG, has driven the overall design.

2.5.1 Purpose and High Level Operation

The purpose of the graphical user interface module is to provide the user a convienent means through which all functional areas of the program may be accessed. The GUI module itself acts as a manager of data elements, where the user is responsible for defining the contents of input data elements and the functional areas are responsible for defining the contents of output data elements.

2.5.2 Accessing

The unit `Display_Menu` is the one initially accessed within the GUI module, and may be called numerous times during program execution. It is the highest level unit, but should not be considered the subprogram driver: the user is!

Inputs

There are no direct inputs to the GUI module.

Outputs

The direct outputs of the GUI module are a set of continuously updated Matlab figure handles and uicontrol object handles which, in total, represent what the user actually sees on the screen. (Refer to *Using MATLAB Graphics* by The Mathworks, Inc. for a comprehensive description of handle graphics.)

Each of these are globally defined, and have variable names which begin with the `H_` prefix. These variables are created in the initialization module, and the data chapter of this document provides additional information.

2.5.3 Global Data

Pre-Defined

All global data created within the Initializiation module is required.

Created for Processing

There are numerous occasions within the GUI code when global data is created for processing. These are for low-level data passing, however, and are necessary due to a Matlab limitation. Specifically, data variables must be defined globally in order to access/modify them within uicontrol callback functions.

2.5.4 Module Decomposition

A complete listing of all units within the GUI module is given in Section 3.2. All of these units are well documented, and further design aspects may be addressed within the code itself.

2.6 Initialization Module

The initialization module contains units which create global data that is used throughout the program. A complete listing of the units comprising the initialization module is given in Section 3.3. The global data initialized by each of these units is discussed in the data chapter of this document.

2.7 Installation Loss Module

Two loss model types are supported by the program: a constant loss model and a non constant loss model. The constant loss model is trivial, in that the difference between uninstalled thrust and installed thrust is always a constant percentage. The non-constant loss model, however, is a more realistic function of flight condition, engine sizing parameters, and engine cycle performance. The calculations performed within this module are consistent with *Aircraft Engine Design* by Mattingly, et. al.

2.7.1 Purpose and High-Level Operation

The purpose of the installation loss module is to reduce the uninstalled thrust value provided by cycle analysis by the effects of inlet drag and nozzle drag. It operates by calculating a loss coefficient for both the inlet and the nozzle, and then combining these with uninstalled thrust to produce an overall installed thrust value.

2.7.2 Accessing

The unit `Apply_Inst_Loss` is the subprogram driver, and is invoked as a Matlab function. All inputs are passed in either as vectors or scalars. All outputs are scalars.

Inputs

The inputs to the unit `Apply_Inst_Loss`, with names as given in the source code, are given in Table 2.4.

Table 2.4: Installation Loss Model Inputs

Variable Name	*Data Type*	*Brief Description*
`ONDES_PIC`	vector	On design PIC vector.
`ONDES_DAT`	vector	On design data vector.
`MAXENG_PIC`	vector	Maximum engine limits PIC vector.
`MAXENG_DAT`	vector	Maximum engine limits data vector.
`MINENG_PIC`	vector	Minimum engine limits PIC vector.
`MINENG_DAT`	vector	Minimum engine limits data.
`OFFDES_PIC`	vector	Off design point PIC vector.
`OFFDES_DAT`	vector	Off design point data vector.
`UN_THRUST`	scalar	Uninstalled thrust.

Outputs

The outputs to the unit `Apply_Inst_Loss`, with names as given in the source code, are given in Table 2.5.

Table 2.5: Installation Loss Model Outputs

Variable Name	*Data Type*	*Brief Description*
`INLET_LC`	scalar	Inlet loss coefficient.
`NOZZLE_LC`	scalar	Nozzle loss coefficient.
`IN_THRUST`	scalar	Installed thrust value.
`MODEL_STATUS`	scalar	Overall pass/fail of loss model.

2.7.3 Global Data

Pre-Defined

Global data with the `C_`, `D_`, and `DLFT_` prefixes are required for processing.

Created for Processing

There is no global data created for processing.

2.7.4 Module Decomposition

A complete listing of units within the installation loss module is given in Section 3.4. Again, these units are well documented and further design issues may be addressed within the code itself.

2.8 Functional Areas Module

The functional areas module serves as the ultimate interface between the GUI module and other modules within the program. Separate drivers for each functional area manage overall processing.

2.8.1 Purpose and High Level Operation

The purpose of the functional areas module is to create the contents of output data elements based on the contents of user-defined input data elements. Each of the functional areas has associated with it an executive unit which extracts the required data from input data elements, performs data initialization necessary to use other modules, executes the core function, and combines all output data into ASCII text.

2.8.2 Accessing

Each functional area is driven by an executive unit which ends with the suffix `_Exec`. Each executive unit is invoked as a Matlab function, where all inputs are passed in as pre-defined global data and all outputs are ASCII text cell arrays.

Inputs

The primary input is the global data structure `S_DATA_STRUCT`.

Outputs

For every case, the output is one or more cell arrays which contain ASCII text.

2.8.3 Global Data

Pre-Defined

The `_Exec` units require the use of global data beginning with the prefixes `C_`, `G_`, `P_`, `S_`, and `SE_`. Refer to the source code and/or the data chapter for further details.

Created for Processing

For the `_Exec` cases which involve an aircraft mission, global `D_` data described in Section 2.11.2 is created for processing.

For the `_Exec` cases which involve optimization, the global parameter `PIC_POINTER` is created to associate PICs with data contained in design vectors. Also for these cases, global data beginning with the prefixes `O_` and `K_` are created for passing information to the objective function. The `O_` data contains objective information, while the `K_` data contains kriging information.

There are other case-specific global data which are created for processing, but not discussed here. Refer to the source code for details.

2.8.4 Module Decomposition

Again, this module acts as an interface between the GUI and other modules within the program. A complete listing of units comprising this module may be found Section 3.5.

2.9 Kriging Module

The kriging module contains the Matlab code necessary for performing pointwise function estimation using a geostatistical procedure known as kriging. It contains a single unit of code, `Kriging.m`, which is exactly the same as that given in *Improving Algorithmic Efficiency of Aircraft Engine Design for Optimal Performance* by Millhouse. Consult this document for further information.

2.10 Lower Level Module

The lower level module contains routines which are fairly general, and not specific to any other module given in this SDD. A complete listing of the units comprising this module is given in Section 3.6. Consult the source code itself for further information.

2.11 Mission Analysis Module

The mission analysis routines used in the program are based on the contents Mattingly's reference. It is important to note that these routines are *independent* of the mission analysis program named OFFX which Mattingly created. Although OFFX is robust and its reuse is highly desirable, the use of a completely different engine cycle model precludes its use. Rather than completely re-engineer the existing OFFX package in order to use TERMAP for cycle analysis, a new set of mission analysis codes has been created. The overall operation is similar to the OFFX package, with many of the same series of calculations being performed. The interface with the engine cycle analysis model is profoundly different, however, and provisions have been introduced to make the mission analysis program more versatile.

2.11.1 Purpose and High Level Operation

The purpose of the mission analysis module is to evaluate combined aircraft and engine performance throughout a given mission profile. It operates by sequentially processing each mission leg, wherein calculations are performed based on the leg type, given data, and permissable constraints.

2.11.2 Accessing

The unit `Mission_Analysis` is the subprogram driver, and is accessed as a Matlab function. All inputs are passed in as pre-defined global data. All outputs are passed out in a data structure.

Inputs

The inputs to the unit Mission_Analysis are passed in as pre-defined global data. All of this data begins with the prefix D_, as given in Table 2.6.

Table 2.6: Mission Analysis Input Data

Variable Name	*Data Type*	*Brief Description*
D_ONDES_T_5	scalar	On design TERMAP T(5) value.
D_ONDES_PIC	vector	On design parameter index codes.
D_ONDES_DAT	vector	On design data.
D_MAXENG_PIC	vector	Maximum engine limits parameter index codes.
D_MAXENG_DAT	vector	Maximum engine limits data.
D_MINENG_PIC	vector	Minimum engine limits parameter index codes.
D_MINENG_DAT	vector	Minimum engine limits data.
D_DRG_TABLE	matrix	Drag table.
D_TO_WEIGHT	scalar	Takeoff weight.
D_NUM_ENGINES	scalar	Number of engines.
D_MAX_LIFT_COEFF	scalar	Maximum lift coefficient.
D_WING_AREA	scalar	Wing area.
D_WING_LOADING	scalar	Wing loading.
D_MACH_CRIT	scalar	Critical Mach number.
D_TO_VEL_RATIO	scalar	Takeoff velocity ratio.
D_TO_FRICT_COEFF	scalar	Takeoff friction coefficient.
D_GEAR_DRAG_COEFF	scalar	Additional friction coefficient for takeoff.
D_THRUST_PCNT_TOL	scalar	Thrust percentage increase tolerance.
D_INITIAL_BETA	scalar	Initial weight fraction.
D_MISS_PROFILE	cell-struct	Mission profile.
D_LEG_FAIL_OPT	scalar	Leg failure option.
D_AB_THROT_TOL	scalar	Afterburner throttle tolerance.
D_AB_THROT_MAX_IT	scalar	Afterburner throttle maximum number of iterations.
D_INST_LOSS	struct	Installation loss model.

Outputs

There is a single output to the unit Mission_Analysis. It is a combined cell array and data structure, where each cell array index corresponds to a mission leg and has structure fields for individual calculations. These output fields are created by the unit Init_Miss_Output.

2.11.3 Global Data

Pre-Defined

In addition to that specified as input parameters, the mission analysis program requires additional global data to be defined prior to accessing. Specifically, all global data beginning with the prefixes `C_`, `E_`, and `LT_` are used. Further information concerning these may be found in the data chapter of this document.

Created for Processing

The parameter `D_MISS_OUTPUT` is the only global parameter created for processing. It is continually updated during processing to assign calculations made for individual mission legs.

2.11.4 Module Decomposition

A complete listing of all units within the mission module is given in Section 3.7.

One basic requirement common to all mission legs is to obtain and use installed engine data. The unit `Get_Off_Design_Data` serves this purpose for all cases, and is itself relatively simple. One of the units it calls, however, is quite complex. Specifically, for the cases where throttling to a thrust is required, the unit `Get_Req_Unin_Data` is responsible for the manipulation of uninstalled TERMAP I/O along with the installation loss model in order to produce uninstalled engine settings that yield a desired installed engine thrust.

2.12 TERMAP I/O Module

The program is designed to use TERMAP in a modified form. In its standard form, TERMAP is intended to be used interactively by a human. In its newly modified form, TERMAP is intended to be used autonomously by a computer. The differences between them strictly deal with its interface and the manner in which it uses external files. The internal processing within TERMAP is unchanged.

2.12.1 Purpose and High Level Operation

The purpose of the TERMAP I/O module is to perform engine cycle analysis using the TERMAP computer program. In its modified state, TERMAP is a stand-alone executable file which reads in the contents of two input ASCII text files and creates a single output ASCII text file. The Matlab routines within this module are responsible for creating the TERMAP input files, invoking TERMAP as a stand-alone operating system executable, and extracting data from the output file.

2.12.2 Accessing

There is no specific driver for this module, although the routines comprising it are normally sequentially accessed.

Inputs

The basic inputs to this module are always the same and are passed in as Matlab row vectors. These vectors collectively define an engine cycle design, maximum engine limits, minimum engine limits, an off design point, the data to be extracted from TERMAP during on design cycle analysis, and the data to be extracted from TERMAP during off design cycle analysis.

Engine Cycle Design The engine cycle design is passed into the module as two Matlab row vectors. One vector contains PIC codes, while the other contains data corresponding to each PIC code. Both of these vectors are assumed to be of the same size, and each must have 25 entries or less. The cycle parameters defined by these vectors have the net effect of *replacing* data in the on design section of the baseline TERMAP input file supplied by the user. As a simple example, suppose that the Matlab commands `CYCLE_PIC = [1  1249]` and `CYCLE_DAT = [10000  0.5]` have been used to define the cycle input PIC and cycle input data vectors, respectively. This would have the net effect of replacing the on design `ALT` (altitude) specification in the baseline TERMAP input file with 10000, and the on design `VEL` (mach number for this case) with 0.5. All other engine cycle design parameters would be used as given in the baseline TERMAP input file.

Maximum Engine Limits The maximum engine limits are passed into the module as two Matlab row vectors. One vector contains PIC codes, while the other contains the maximum limits corresponding to each PIC code. Both of these vectors are assumed to be of the same size, and each must have 10 entries or less. The maximum limit parameters defined by these vectors have the net effect of *introducing* data in the on design section of the baseline TERMAP input file supplied by the user. (Recall that it is an explicit requirement that the baseline TERMAP input file not contain any maximum limit specifications whatsoever.) As a simple example, suppose that the Matlab commands `MAX_PIC = [800  932  1151]` and `MAX_DAT = [1500  1.2  2260]` have been used to define the maximum limit input PIC and maximum limit input data vectors, respectively. This would have the net effect of introducing the necessary TERMAP constraint variables (`ENGMAX` and `MAXLIM`) in the baseline TERMAP input file to require `P(4)` to be less than 1500, `RMIX(1)` to be less than 1.2, and `T(4)` to be less than 2260.

Minimum Engine Limits The minimum engine limits are passed into the module as two Matlab row vectors. One vector contains PIC codes, while the other contains the minimum limits corresponding to each PIC code. Both of these vectors are assumed to be of the same size, and each must have 5 entries or less. The minimum limit parameters defined by these vectors have the net effect of *introducing* data in the on design section of the baseline TERMAP input file supplied by the user. (Recall that it is an explicit requirement that the baseline TERMAP input file not contain any minimum limit specifications whatsoever.) As a simple example, suppose that the Matlab commands `MIN_PIC = [1932]` and `MIN_DAT = [0.8]` have been used to define the minimum limit input PIC and minimum limit input data vectors, respectively. This would have the net effect of introducing the necessary TERMAP constraint variables (`ENGMIN` and `MINLIM`) in the baseline TERMAP input file to require `RMIX(1)` to be greater than 0.8.

TERMAP On Design Data Extraction The desired on design data parameters to be extracted from TERMAP are passed into the module as a single Matlab row vector. It must have 20 entries or less, and only contains PIC codes. As a simple example, suppose that the Matlab command `ONDES_OUT = [764 657]` has been used to define this vector. This would cause the modified version of TERMAP to write the on design values for `IFAIL` and `FN` to the on design section of the output file.

TERMAP Off Design Data Extraction The desired off design data parameters to be extracted from TERMAP are passed into the module as a single Matlab row vector. It must have 20 entries or less, and only contains PIC codes. As a simple example, suppose that the Matlab command `OFFDES_OUT = [764 657]` has been used to define this vector. This would cause the modified version of TERMAP to write the off design values for `IFAIL` and `FN` to the off design section of the output file.

Outputs

There is a single output to this module. It is a single Matlab vector which always contains 40 entries. The first 20 are data values extracted from TERMAP during off design evaluation, while the second 20 are data values extracted from TERMAP during on design evaluation. *The on design values are actually created first within TERMAP, but are located in the latter half of this output vector nonetheless.* The TERMAP extraction vectors described in the previous two sections are used for determining the correspondence between data values and PIC codes. Refer to the source code for good examples on how the overall I/O is accomplished.

In the event of a physical TERMAP crash, all numerical data in the output vector will be set to the number defined globally as `E_TERMAP_CRASH`. The ability to recover from a crash is absolutely critical, particularly for engine optimization runs: the number of individual TERMAP calls could easily be in the thousands.

2.12.3 Global Data

Pre-Defined

The variables `P_TERMAP_PATH` and `P_TERMAP_EXE` are required for processing. Refer to the data chapter for a description of these parameters.

Created for Processing

There is no global data created for processing.

2.12.4 Module Decomposition

A complete listing of all units within the TERMAP I/O module is given in Section 3.8. These units are sequentially accessed in the following order.

1. The unit `Create_Main_Input` is called to copy the user defined baseline TERMAP input file to the TERMAP I/O directory. The new file is named `INPUT_FILE_1`. *Note: This is only needed once.*

2. The unit `Gen_Termap_Vec` is called to map the input vectors described in Section 2.12.2 to a single, 170 element vector. All unused parameters are mapped to null entries, with unused PIC codes assigned a value of -1 and unused data assigned a value of 0.

3. The unit `Write_Termap_Infile` is called to write the 170 element vector to a file named `INPUT_FILE_2` in the TERMAP I/O directory.

4. The unit `Invoke_Termap` is called to execute the modified version of TERMAP at the operating system level.

5. The unit `Did_Termap_Crash` is called to determine if TERMAP crashed.

6. If a crash did not occur, then the unit `Get_Ter_Out_40` is called to read in the contents of the newly created TERMAP output file, named `OUTPUT_FILE`, which contains 40 elements.

Further details encompassing the overall TERMAP interface with external files are not given in this document. U.S. Government personnel may find this information on a corresponding DoD-Limited CD-ROM.

Chapter 3

Unit Listing

Overview

The purpose of this section is to provide a complete listing of all source code unit names comprising the Integrated Multi-Objective Multi-Disciplinary Jet Engine Design Optimization Program. The unit names are organized by the modules described in the SDD.

3.1 Genetic Algorithm

The units comprising the genetic algorithm module are given in Table 3.1.

Table 3.1: Genetic Algorithm Units

Exact Name	*Long Name - Purpose*
`Arith_Crossover`	Arithmetic Crossover - To create two child designs from two parent designs by performing a linear interpolation between them.
`Boundary_Mutation`	Boundary Mutation - To randomly change one of the parameters of a design to either the upper or lower bound.
`Delta`	Delta - To return the amount of change for a design variable.
`Genetic_Algorithm`	Genetic Algorithm - To perform function optimization using genetic algorithms.
`Heuristic_Crossover`	Heuristic Crossover - To create two child designs from two parent designs by extrapolating in the direction of the better parent.
`Initialize`	Initialize - To create an initial population of designs for use with a genetic algorithm optimizer.
`Multi_Nonunif_Mutation`	Multi-Parameter Nonuniform Mutation - To change all of the parameters of a design based on a non-uniform probability distribution.
`Nonunif_Mutation`	Nonuniform Mutation - To change one of the parameters of a design based on a non-uniform probability distribution.
`Norm_Geom_Select`	Normalized Geometric Distribution Selection - To select designs from an old population to carry over to a new population based on the normalized geometric distribution.
`Rand_Choose`	Random Choose - To randomly choose one of two parameters.
`Rand_Choose_Idx`	Random Choose Index - To randomly choose an index into a vector.
`Simple_Crossover`	Simple Crossover - Two create two child designs from two parent designs by swapping the design variables before and after a randomly chosen position.

Genetic Algorithm Units - continued

Exact Name	*Long Name - Purpose*
`Unif_Mutation`	Uniform Mutation - To change one of the parameters of a design based on a uniform probability distribution.

3.2 Graphical User Interface

The units comprising the graphical user interface module are given in the Table 3.2.

Table 3.2: Graphical User Interface Units

Exact Name	*Long Name - Purpose*
`Add_To_Messages`	Add To Messages - To add an operator message for display.
`Check_For_Errors`	Check For Errors - To check user data for possible errors.
`Clear_Menu`	Clear Menu - To clear data within a GUI menu.
`Disp_PIC_Str`	Display PIC String - To display the paramter index code / TERMAP variable name correlation.
`Display_Menu`	Display Menu - To display and activate a GUI menu.
`Display_Output`	Display Output - To display output text on the screen.
`Edit_Element`	Edit Element - To edit a data element.
`Execute_Prog`	Execute Program - To execute a program corresponding to the displayed menu.
`Exit_Program`	Exit Program - To exit the program.
`File_For_Open`	File For Open - To obtain the filename and pathname of a user selected file.
`File_For_Save`	File For Save - To obtain the filename and pathname for a user defined file.
`Finish_UI_Data`	Finish User Interface Data - To set global data indicating an update to the global data structure is required.
`Gen_Status_Str`	Generate Status String - To generate a cell array of strings which represents the data element status for a particular menu.
`Get_AC_Sizing`	Get Aircraft Sizing Parameters - To get the sizing configuration of a scaleable aircraft from the user.
`Get_Inst_Loss`	Get Installation Loss Model - To get the installation loss model configuration from the user.
`Get_Kriging`	Get Kriging Usage - To get an overall use or don't use.

Graphical User Interface Units - continued

Exact Name	*Long Name - Purpose*
`Get_Miss_Profile`	Get Mission Profile - To get a mission profile from the user.
`Get_Obj_WO_Miss`	Get Objectives Without Mission - To get a design objective configuration without mission from the user.
`Get_PIC_Data_Pairs`	Get PIC Data Pairs - To get parameter index code / data pairs from the user.
`Get_PIC_Data_Table`	Get PIC Data Table - To get a table of parameter index codes and data from the user.
`Get_PIC_Values`	Get PIC Values - To get a list of parameter index codes from the user.
`Get_Standard_Data`	Get Standard Data - To get data from the user.
`Invoke_Text_Edit`	Invoke Text Editor - To invoke a text editor to allow the user to edit text files.
`Leg_Num_Callback`	Leg Number Callback - To set mission data corresponding to a leg number.
`Leg_Type_Callback`	Leg Type Callback - To set mission data corresponding to a leg type.
`Load_Data_2`	Load Data 2 - To transfer data previously loaded from a *.mat file into the program's current menu structure.
`Load_GUI_Data`	Load GUI Data - To load GUI data from a file.
`Mat_File_Load`	Mat File Load - To select the filename for loading desired data elements from a *.mat file.
`Mat_File_Save`	Mat File Save - To select the filename for saving desired data elements to a *.mat file.
`Reset_Element`	Reset Element - To reset a data element to its default configuration.
`Rtn_To_Main_Menu`	Return To Main Menu - To return the GUI to the main menu.
`Save_Data`	Save Data - To save desired data elements to a *.mat file.
`Save_GUI_Data`	Save GUI Data - To save GUI data for a particular menu.
`Sel_Data_Load`	Select Data Load - To allow the user to select existing data elements from a data file.
`Sel_Data_Save`	Select Data Save - To get selected data elements for file save from the user.
`Set_Data_Vals`	Set Data Vals - To display the data and values corresponding to a particular leg type selection.
`Size_Type_Callback`	Size Type Callback - To display the data and values corresponding to a particular A/C sizing type scheme.

Graphical User Interface Units - continued

Exact Name	*Long Name - Purpose*
`Update_Leg_Data`	Update Leg Data - To update the mission profile after user editing.

3.3 Initialization

The units comprising the initialization module are given in the Table 3.3.

Table 3.3: Initialization Unit Listing

Exact Name	*Long Name - Purpose*
`Gen_PIC_Str_Display`	Generate PIC Strings For Display - To create the strings relating parameter index codes to TERMAP variable names.
`Init_Data_Struct`	Initialize Data Structure - To initialize the global data element structure.
`Init_Errors`	Initialize Errors - To initialize the errors which can be captured within the GUI prior to interaction with other modules.
`Init_Graphics`	Initialize Graphics - To initialize graphics data.
`Init_Kriging_Data`	Initialize Kriging Data - To initialize constants and kriging model parameters.
`Init_Leg_Mapping`	Initialize Leg Mapping - To initialize the relationship between each mission leg type and its required input data.
`Init_Leg_Strings`	Initialize Leg Strings - To initialize the strings associated with the name of each leg type.
`Init_Leg_Types`	Initialize Leg Types - To initialize mission leg types.
`Init_Loss_Types`	Initialize Loss Types - To initialize the installation loss types.
`Init_Messages`	Initialize Messages - To initialize the user-messages which can be displayed by the graphical user interface.
`Init_Miss_Constants`	Initialize Mission Constants - To initialize the global data which is primarily used for mission analysis.
`Init_PIC_Str`	Initialize Parameter Index Code Strings - To associate character strings for each TERMAP parameter index code.
`Init_Sizing_Types`	Initialize Sizing Types - To initialize scaleable aircraft sizing types.
`Main`	Main - To initialize all global data and data structures used by the program, and to pass control to the graphical user interface.

3.4 Installation Loss

The units comprising the installation loss module are given in Table 3.4.

Table 3.4: Installation Loss Module Unit Listing

Exact Name	*Long Name - Purpose*
`Apply_Inst_Loss`	Apply Installation Losses
`Calc_IMS`	Calculate Integral Mean Slope
`Calc_Sup_Nozzle_Drag`	Calculate Subsonic Nozzle Drag Coefficient
`Calc_Total_Press`	Calculate Total Pressure
`Calc_Total_Temp`	Calculate Total Temperature
`Gen_Loss_Tabs`	Generate Non-Constant Installation Loss Drag Tables
`Get_Sub_Inlet_Loss`	Get Subsonic Inlet Loss Coefficient
`Get_Sub_Nozzle_Loss`	Get Subsonic Nozzle Loss Coefficinet
`Get_Sup_Inlet_Loss`	Get Supersonic Inlet Loss Coefficient
`Get_Sup_Nozzle_Loss`	Get Subsonic Nozzle Loss Coefficient
`Get_Tra_Nozzle_Loss`	Get Transonic Nozzle Loss Coefficient

3.5 Functional Areas

The units comprising the functional areas module are given in Table 3.5.

Table 3.5: Functional Areas Unit Listing

Exact Name	*Long Name - Purpose*
`EOWIM_FAC_Exec`	Engine Optimization With Mission Fixed Aircraft Executive - To perform engine optimization with mission for a fixed aircraft.
`EOWIM_FAC_Fcn`	Engine Optimization With Mission Fixed Aircraft Objective Function - To determine a given design's objective function value as part of the optimization process.
`EOWIM_SAC_Exec`	Engine Optimization With Mission Scaleable Aircraft Executive - To perform engine optimization with mission for a scaleable aircraft.
`EOWIM_SAC_Fcn`	Engine Optimization With Mission Scaleable Aircraft Objective Function - To determine a given design's objective function value as part of the optimization process.
`EOWOM_Exec`	Engine Optimization Without Mission Executive - To perform engine optimization without mission.
`EOWOM_Fcn`	Engine Optimization Without Mission Objective Function - To determine a given design's objective function value as part of the optimization process.

Functional Areas Unit Listing - continued

Exact Name	*Long Name - Purpose*
`SACA_Exec`	Stand Alone Engine Cycle Analysis Executive - To perform stand alone engine cycle analysis.
`SAMA_Exec`	Stand Alone Mission Analysis Executive - To perform stand alone mission analysis.

3.6 Lower Level

The units comprising the lower level module are given in the Table 3.6.

Table 3.6: Lower Level Unit Listing

Exact Name	*Long Name - Purpose*
`Calc_Scaled_Obj`	Calculate Scaled Objective - To determine the scaled objective value for a design objective.
`Calc_Takeoff_Weight`	Calculate Takeoff Weight - To calculate the takeoff weight of an aircraft as a function of fuel weight and sizing parameters.
`Change_To_Dir`	Change To Directory - To change the current Matlab working directory.
`Compact_Design_Var`	Compact Design Variables - To compact a design variable configuration by removing unused entries.
`Compact_Engine_Spec`	Compact Engine Specification - To compact an engine specification by removing unused entries.
`Delete_File`	Delete File - To delete a file.
`Get_File_Size`	Get File Size - To determine the size of a file in bytes.
`Linear_Interp`	Linear Interpolation - To perform linear interpolation for data within a matrix data table.
`Load_Data`	Load Data - To transform the contents of a file into a matrix.
`Map_Box_To_Pos`	Map Box To Position - To map two x-y coordinate pairs into a Matlab position vector.
`Map_Coords`	Map Coordinates - To map x-y coordinate pairs defined in a reference domain to x-y coordinates defined in another domain.
`Mat_To_Cell`	Matrix To Cell Array - To create a cell array of strings corresponding to a matrix.
`Write_Mat_To_File`	*Development - need to move*

3.7 Mission

The units comprising the mission module are given in Table 3.7.

Table 3.7: Mission Unit Listing

Exact Name	*Long Name - Purpose*
`Assn_Leg_Out_Data`	Assign Leg Output Data - To assign output data for a mission leg.
`Best_Cruise_Mach`	Best Cruise Mach - To evaluate a best cruise mach / best cruise altitude mission leg.
`Calc_DG_BCMBCA_Leg`	To calculate the delta-gamma product for a best cruise mach / best cruise altitude mission leg.
`Calc_DP_LOIT_Leg`	To calculate the drag parameter for a loiter mission leg.
`Calc_Drag_Coeff`	To calculate the drag coefficient of an object.
`Calc_Drag_Force`	To calculate the drag force acting on an object.
`Calc_Dyn_Press`	To calculate the dynamic pressure induced by flight velocity at a given altitude.
`Calc_FW_BCMBCA_Leg`	To calculate the final weight of an A/C after a best cruise mach / best cruise altitude mission leg.
`Calc_FW_CA_Leg`	To calculate the final weight of an A/C after a climb and acceleration leg.
`Calc_FW_CASC_Leg`	To calculate the final weight of an A/C after a constant altitude/speed cruise mission leg.
`Calc_FW_CAST_Leg`	To calculate the final weight of an A/C after a constant altitute/speed turn mission leg.
`Calc_FW_CEHM_Leg`	To calculate the final weight of an A/C after a constant energy height maneuver mission leg.
`Calc_FW_CSC_Leg`	To calculate the final weight of an A/C after a constant speed climb mission leg.
`Calc_FW_HA_Leg`	To calculate the final weight of an A/C after a horizontal acceleration mission leg.
`Calc_FW_LOIT_Leg`	To calculate the final weight of an A/C after a loiter mission leg.
`Calc_FW_TA_Leg`	To calculate the final weight of an A/C after the acceleration portion of a takeoff mission leg.
`Calc_FW_TR_Leg`	To calculate the final weight of an A/C after the rotation portion of a takeoff mission leg.
`Calc_FW_WU_Leg`	To calculate the final weight of an A/C after a warmup mission leg.
`Calc_Lift_Coeff`	To calculate the lift coefficient of an A/C.
`Calc_Stall_Vel`	To calculate the minimum speed at which flight is possible for an A/C.
`Climb_And_Accel`	Climb And Acceleration - To evaluate a climb and acceleration mission leg.
`Const_Alt_Spd_Crs`	Constant Altitude/Speed Cruise - To evaluate a constant altitude/speed cruise mission leg.

Mission Unit Listing - continued

Exact Name	*Long Name - Purpose*
`Const_Alt_Spd_Turn`	Constant Altitude/Speed Turn - To evaluate a constant altitude/speed turn mission leg.
`Const_EH_Man`	Constant Energy Height Maneuver - To evaluate a constant energy height maneuver mission leg.
`Const_Spd_Clmb`	Constant Speed Climb - To evaluate a constant speed climb mission leg.
`Convert_Mach_To_Vel`	Convert Mach Number To Velocity - To convert a mach number at a given altitude to a velocity.
`Deliv_Exp`	Deliver Expendables - To evaluate a deliver expendables mission leg.
`Gen_Atm_Table`	Generate Atmospheric Data Table - To generate a data table which contains atmospheric properties as a function of altitude.
`Gen_Drg_Table`	Generate Drag Table - To generate a drag table for an object.
`Get_Off_Design_Data`	Get Off Design Data - To determine off design engine performance given an engine design and an off design point.
`Get_Req_Unin_Data`	Get Required Uninstalled Data - To determine the required uninstalled engine settings that should be used to satisfy an installed thrust requirement
`Horizontal_Accel`	Horizontal Acceleration - To evaluate a horizontal acceleration mission leg.
`Init_Leg_Out_Data`	To initialize output data for a mission leg.
`Init_Miss_Output`	To initialize all output data for mission analysis.
`Loiter`	Loiter - To evaluate a loiter mission leg.
`Mission_Analysis`	Mission Analysis - To perform A/C mission analysis.
`Proc_Miss_Leg`	Process Mission Leg - To process a mission leg.
`Takeoff`	Takeoff - To evaluate a takeoff mission leg.
`Throttle_AB`	Throttle Afterburner - To throttle the afterburner setting in order to attain a desired thrust.
`Unused_Leg`	Unused Leg - To evaluate an unused mission leg.
`Warm_Up`	Warm Up - To evaluate a warm up mission leg.

3.8 TERMAP I/O

The units comprising the TERMAP I/O module are given in Table 3.8.

Table 3.8: TERMAP I/O Unit Listing

Exact Name	*Long Name - Purpose*
`Create_Main_Input`	To create a copy of the user defined primary TERMAP input file in the directory which contains the TERMAP executable.
`Did_Termap_Crash`	Did TERMAP Crash - To determine whether or not TERMAP physically crashed during execution.
`Gen_Termap_Vec`	Generate TERMAP Vector - To assemble segments of an engine design into a single vector which, after further processing, can be fed into TERMAP.
`Get_Ter_Out_40`	Get TERMAP Output With 40 Entries - To read in numerical values from the TERMAP output data file which contains 40 elements.
`Invoke_Termap`	Invoke TERMAP - To call TERMAP for engine cycle analysis.
TERMAP Executable	The TERMAP executable file is user supplied. Refer to the SUG and SDD for details.
`Write_Termap_Infile`	Write TERMAP Input File - To create the TERMAP second input file in a form suitable for reading by TERMAP.

3.9 Kriging

The kriging module contains a single unit of code, `Kriging.m`. Refer to *Improving Algorithmic Efficiency of Aircraft Engine Design for Optimal Performance* by Millhouse for any details.

Chapter 4

Data Listing

Overview

The purpose of this section is to provide a complete listing of all global data created within the initialization module of the Integrated Multi-Objective Multi-Disciplinary Jet Engine Design Optimization Program. The data given in this section is organized by the unit used to create it.

4.1 Init Graphics

The global data defined by this unit is given in Table 4.1.

Table 4.1: Global Data Created by Init Graphics

Name	*Data Type*	*Brief Description*
`H_FIG_OPMSG`	figure	The operator message window figure graphics handle.
`H_FIG_OPTIONS`	figure	The options window graphics handle.
`H_FIG_STATUS`	figure	The status window graphics handle.
`H_FIG_DATA`	figure	The data window graphics handle.
`H_UI_OPT_1`	uicontrol	The first option graphics handle.
`H_UI_OPT_2`	uicontrol	The second option graphics handle.
`H_UI_OPT_3`	uicontrol	The third option graphics handle.
`H_UI_OPT_4`	uicontrol	The fourth option graphics handle.
`H_UI_OPT_5`	uicontrol	The fifth option graphics handle.
`H_UI_OPT_6`	uicontrol	The sixth option graphics handle.
`H_UI_OPT_7`	uicontrol	The seventh option graphics handle.
`H_UI_OPMSG`	uicontrol	The operator message graphics handle.
`H_UI_STATUS`	uicontrol	The status graphics handle.
`H_UI_DATA`	uicontrol	The data graphics handle.
`G_NUM_MENUS`	scalar	The total number of menus.
`G_CURRENT_MENU`	scalar	The currently displayed menu. Initially set to `G_PRIMARY_MENU`.
`G_PRIMARY_MENU`	scalar	The reference number for the primary menu.
`G_SACA_MENU`	scalar	The reference number for the stand alone cycle analysis menu.
`G_SAMA_MENU`	scalar	The reference number for the stand alone mission analysis menu.
`G_EOWIM_FAC_MENU`	scalar	The reference number for the engine optimization with mission for a fixed aircraft menu.
`G_EOWIM_SAC_MENU`	scalar	The reference number for the engine optimization with mission for a scaleable aircraft menu.
`G_EOWOM_MENU`	scalar	The reference number for the engine optimization without mission menu.

Global Data Created by Init Graphics - continued

Name	*Data Type*	*Brief Description*
G_ADD_FCNS_MENU	scalar	The reference number for the additional functions menu.
G_OPMSG_FIG_TITLE	cell	A cell array of strings. Each index corresponds to a menu, and contains the text string to be displayed as the title within the operator message window.
G_OPTIONS_FIG_TITLE	cell	A cell array of strings. Each index corresponds to a menu, and contains the text string to be displayed as the title within the options figure.
G_STATUS_FIG_TITLE	cell	A cell array of strings. Each index corresponds to a menu, and contains the text string to be displayed as the title within the status figure.
G_DATA_FIG_TITLE	cell	A cell array of strings. Each index corresponds to a menu, and contains the text string to be displayed as the title within the data figure.
G_OPT_STR_1	cell	A cell array of strings. Each index corresponds to a menu, and contains the text string to be displayed for the first option.
G_OPT_FCN_1	cell	A cell array of strings. Each index corresponds to a menu, and contains the Matlab commands or unit to be invoked upon selection of the first option.
G_OPT_STR_2	cell	A cell array of strings. Each index corresponds to a menu, and contains the text string to be displayed for the second option.
G_OPT_FCN_2	cell	A cell array of strings. Each index corresponds to a menu, and contains the Matlab commands or unit to be invoked upon selection of the second option.
G_OPT_STR_3	cell	A cell array of strings. Each index corresponds to a menu, and contains the text string to be displayed for the third option.
G_OPT_FCN_3	cell	A cell array of strings. Each index corresponds to a menu, and contains the Matlab commands or unit to be invoked upon selection of the third option.
G_OPT_STR_4	cell	A cell array of strings. Each index corresponds to a menu, and contains the text string to be displayed for the fourth option.

Global Data Created by Init Graphics - continued

Name	*Data Type*	*Brief Description*
G_OPT_FCN_4	cell	A cell array of strings. Each index corresponds to a menu, and contains the Matlab commands or unit to be invoked upon selection of the fourth option.
G_OPT_STR_5	cell	A cell array of strings. Each index corresponds to a menu, and contains the text string to be displayed for the fifth option.
G_OPT_FCN_5	cell	A cell array of strings. Each index corresponds to a menu, and contains the Matlab commands or unit to be invoked upon selection of the fifth option.
G_OPT_STR_6	cell	A cell array of strings. Each index corresponds to a menu, and contains the text string to be displayed for the sixth option.
G_OPT_FCN_6	cell	A cell array of strings. Each index corresponds to a menu, and contains the Matlab commands or unit to be invoked upon selection of the sixth option.
G_OPT_STR_7	cell	A cell array of strings. Each index corresponds to a menu, and contains the text string to be displayed for the seventh option.
G_OPT_FCN_7	cell	A cell array of strings. Each index corresponds to a menu, and contains the Matlab commands or unit to be invoked upon selection of the seventh option.
G_DEF_OPMSG_STR	cell	A cell array of strings. Each index corresponds to a menu, and contains the default message to be displayed in the operator message window.
G_DEF_DATA_STR	cell	A cell array of strings. Each index corresponds to a menu, and contains the default information to be displayed in the data window.
G_FONTS	cell-struct	A combined cell array and data structure containg font information. Each index corresponds to a menu.
G_ORIG_11_PT	scalar	A font size to be used for GUI scaling on multiple platforms.
G_ORIG_10_PT	scalar	A font size to be used for GUI scaling on multiple platforms.
G_ORIG_9_PT	scalar	A font size to be used for GUI scaling on multiple platforms.

Global Data Created by Init Graphics - continued

Name	*Data Type*	*Brief Description*
U_UPDATE	scalar	Reference number indicating that the user has completed a process and an update to the global data structure is required.
U_NO_UPDATE	scalar	Reference number indicating that the user has canceled a process and no update to the global data structure is required.

4.2 Init Sizing Types

The global data defined by this unit is given in Table 4.2.

Table 4.2: Global Data Created by Init Sizing Types

Name	*Data Type*	*Brief Description*
SZT_NUM_TYPES	scalar	The total number of sizing types supported.
SZT_LINEAR	scalar	Reference number for the linear sizing type.
SZT_NONLINEAR	scalar	Reference number for the nonlinear sizing type.
SZT_COEFF_PER_TYPE	vector	Vector containing the number of coefficients for each sizing type.

4.3 Init Loss Types

The global data defined by this unit is given in Table 4.3.

Table 4.3: Global Data Created by Init Sizing Types

Name	*Data Type*	*Brief Description*
ILT_CONST	scalar	Reference number for a constant installation loss model.
ILT_NON_CONST	scalar	Reference number for a non-constant installation loss model.

4.4 Init Data Struct

The global data defined by this unit is given in Table 4.4.

Table 4.4: Global Data Created by Init Data Struct

Name	*Data Type*	*Brief Description*
S_NUM_ELEMENTS	scalar	The total number of data element types.

Global Data Created by Init Data Struct - continued

Name	*Data Type*	*Brief Description*
SE_TER_IN_FILE	scalar	Data element reference number for a TERMAP input file.
SE_TER_IN_TYPE	scalar	Data element reference number for the type of a TERMAP input file. *This is currently unused.*
SE_ENG_ONDES	scalar	Data element reference number for an engine on design specification.
SE_ENG_MAXENG	scalar	Data element reference number for an engine maximum limit specification.
SE_ENG_MINENG	scalar	Data element reference number for an engine minimum limit specification.
SE_OFF_PT	scalar	Data element reference number for an off design point specification.
SE_DES_ON_OUT	scalar	Data element reference number for desired on design outputs.
SE_DES_OFF_OUT	scalar	Data element reference number for desired off design outputs.
SE_AC_DRAG_FILE	scalar	Data element reference number for an aircraft drag file.
SE_AC_CONSTANTS	scalar	Data element reference number for aircraft constants.
SE_AC_TO_WEIGHT	scalar	Data element reference number for aircraft takeoff weight.
SE_AC_VAR_SIZE	scalar	Data element reference number for a scaleable aircraft sizing configuration.
SE_MISS_PROFILE	scalar	Data element reference number for a mission profile.
SE_MISS_CONSTANTS	scalar	Data element reference number for mission constants.
SE_INST_LOSS	scalar	Data element reference number for an installation loss model configuration.
SE_DES_VAR	scalar	Data element reference number for a design variable configuration.
SE_DES_OBJ_WIM	scalar	Data element reference number for a design objectives with mission configuration.
SE_DES_OBJ_WOM	scalar	Data element reference number for a design objectives without mission configuration.
SE_GA_OPTS	scalar	Data element reference number for genetic algorithm options.
SE_GA_END_POP	scalar	Data element reference number for genetic algorithm end population.

Global Data Created by Init Data Struct - continued

Name	*Data Type*	*Brief Description*
SE_GA_BEST_POP	scalar	Data element reference number for genetic algorithm best population,
SE_GA_TRACE_INFO	scalar	Data element reference number for genetic algorithm trace information.
SE_OPT_DES_VAR	scalar	Data element reference number for genetic algorithm best solution.
SE_CYCLE_ANALYSIS	scalar	Data element reference number for cycle analysis output.
SE_MISS_ANALYSIS	scalar	Data element reference number for mission analysis output.
SE_KRIGING	scalar	Data element reference number for kriging usage.
S_NUM_EDIT_TYPES	scalar	The total number of data element edit types.
ST_PICS	scalar	Reference number for editing a data element consisting of parameter index codes only.
ST_PIC_DATA_PRS	scalar	Reference number for editing a data element consisting of parameter index codes and a single parameter for each.
ST_PIC_DATA_TAB	scalar	Reference number for editing a data element consisting of parameter index codes and multiple parameters for each.
ST_MISS_PROFILE	scalar	Reference number for editing the mission profile data element.
ST_STANDARD	scalar	Reference number for editing a data element consisting of numerical data only.
ST_FILE_INPUT	scalar	Reference number for editing a data element consisting of a file specification.
ST_DISP_STRING	scalar	Reference number for viewing a data element comprised of ASCII text strings.
ST_TER_IN_TYPE	scalar	*Currently unused.*
ST_AC_VAR_SIZE	scalar	Reference number for editing the scaleable aircraft data element.
ST_OBJ_WOM	scalar	Reference number for editing the design objectives without mission data element.
ST_INST_LOSS	scalar	Reference number for editing the installation loss model data element.
ST_KRIGING	scalar	Reference number for editing the kriging usage data element.
S_MEN_DATA_MAP	cell	The mapping between menus and the data which it contains. Each cell array index corresponds to a menu, and contains a vector of data element reference numbers.

Global Data Created by Init Data Struct - continued

Name	*Data Type*	*Brief Description*
S_DATA_STRUCT	cell-struct	The global data structure is used by the graphical user interface. It is a two dimensional cell array, where the first index corresponds to menu and the second index corresponds to data element position. The data available into this 2D cell array is a data structure with multiple fields containing data element information.
DEF_DATA_STRUCT	cell-struct	The default global data structure.

4.5 Init Leg Types

The global data defined by this unit is given in Table 4.5.

Table 4.5: Global Data Created by Init Leg Types

Name	*Data Type*	*Brief Description*
L_NUM_TYPES	scalar	The total number of mission leg types.
LT_UNUSED	scalar	Reference number for an unused mission leg.
LT_CSC_MINANG	scalar	Reference number for a constant speed climb mission leg with a minimum climb angle limitation.
LT_CSC_MAXDIST	scalar	Reference number for a constant speed climb mission leg with a maximum distance limitation.
LT_CSC_MINRATE	scalar	Reference number for a constant speed climb mission leg with a minimum climb rate limitation.
LT_CSC_MAXTIME	scalar	Reference number for a constant speed climb mission leg with a maximum time limitation.
LT_HA_MAXDIST	scalar	Reference number for a horizontal acceleration mission leg with a maximum distance limitation.
LT_HA_MAXTIME	scalar	Reference number for a horizontal acceleration mission leg with a maximum time limitation.
LT_CA_MINANG	scalar	Reference number for a climb and acceleration mission leg with a minimum climb angle limitation.
LT_CA_MAXDIST	scalar	Reference number for a climb and acceleration mission leg with a maximum distance limitation.

Global Data Created by Init Leg Types - continued

Name	*Data Type*	*Brief Description*
LT_CA_MINRATE	scalar	Reference number for a climb and acceleration mission leg with a minimum climb rate limitation.
LT_CA_MAXTIME	scalar	Reference number for a climb and acceleration mission leg with a maximum time limitation.
LT_TO	scalar	Reference number for a takeoff mission leg.
LT_CASC_DIST	scalar	Reference number for a constant altitude speed cruise mission leg with a distance specification.
LT_CASC_TIME	scalar	Reference number for a constant altitude speed cruise mission leg with a time specification.
LT_CAST	scalar	Reference number for a constant altitude speed turn mission leg.
LT_BCMBCA_DIST	scalar	Reference number for a best cruise mach best cruise altitude mission leg with a distance specification.
LT_BCMBCA_TIME	scalar	Reference number for a best cruise mach best cruise altitude mission leg with a time specification.
LT_LOIT_DIST	scalar	Reference number for a loiter mission leg with a distance specification.
LT_LOIT_TIME	scalar	Reference number for a loiter mission leg with a time specification.
LT_WU	scalar	Reference number for a warmup mission leg.
LT_CEHM	scalar	Reference number for a constant energy height maneuver mission leg.
LT_DELX	scalar	Reference number for a deliver expendables mission leg.

4.6 Init Leg Mapping

The global data defined by this unit is given in Table 4.6.

Table 4.6: Global Data Created by Init Leg Mapping

Name	*Data Type*	*Brief Description*
D_NUM_INDICES	scalar	Total number of mission data indices.
DX_ALT	scalar	Reference index for altitude.
DX_INIT_ALT	scalar	Reference index for initial altitude.
DX_FIN_ALT	scalar	Reference index for final altitude.
DX_MACH	scalar	Reference index for mach number.

Global Data Created by Init Leg Mapping - continued

Name	*Data Type*	*Brief Description*
DX_INIT_MACH	scalar	Reference index for initial mach number.
DX_FIN_MACH	scalar	Reference index for final mach number.
DX_DIST	scalar	Reference index for distance.
DX_DIST_MAX	scalar	Reference index for maximum distance.
DX_MIN_CLMBANG	scalar	Reference index for minimum climb angle.
DX_MIN_CLMBRATE	scalar	Reference index for minimum climb rate.
DX_TIME	scalar	Reference index for time.
DX_MAX_TIME	scalar	Reference index for maximum time.
DX_ROT_TIME	scalar	Reference index for rotation time.
DX_ABSET	scalar	Reference index for afterburner setting.
DX_ABOPT	scalar	Reference index for afterburner option.
DX_TURNS	scalar	Reference index for number of turns.
DX_LOAD_FCTR	scalar	Reference index for load factor.
DX_VERT_FRACT	scalar	Reference index for vertical fraction.
DX_DELIV_EXP	scalar	Reference index for expendable weight to be delivered.
TYPE_DATA_MAP	cell	The mapping between mission leg types and the required data for them. It is a cell array, where each index corresponds to a leg type and contains a vector of the reference data indices given above.

4.7 Init Messages

This unit creates a variety of message strings for display. Each message variable begins with the prefix M_. Refer to the actual source code for details.

4.8 Init PIC Str

This unit creates the variable P_PIC_STR. It is a cell array with 2288 entries, where each index corresponds to a TERMAP PIC. Each entry contains the TERMAP variable name according to the entry number PIC. For example, P_PIC_STR{1039} = 'SMRELL12';

4.9 Gen PIC Str Display

This unit creates the variable P_PIC_STR_DISPLAY. This is a cell array of ASCII text which the user sees in the data window upon selecting the PIC/Variable button. Refer to the source code for details.

4.10 Init Errors

global data defined by this unit is given in Table 4.7.

Table 4.7: Global Data Created by Init Errors

Name	*Data Type*	*Brief Description*
E_TEST_PASS	scalar	Signifies an overall pass of testing.
E_TEST_FAIL	scalar	Signifies an overall fail of testing.
E_USER_DEFINED	scalar	Signifies that one or more data elements are undefined.
E_CONSISTENT	scalar	Signifies that one or more data elements are inconsistent.
EC_TER_IN_FILE	scalar	Inconsistent TERMAP input file.
EC_TER_IN_TYPE	scalar	*Unused*
EC_ENG_ONDES	scalar	Signifies an inconsistent engine on design definition.
EC_ENG_MAXENG	scalar	Signifies an inconsistent engine maximum limits definition.
EC_ENG_MINENG	scalar	Signifies an inconsistent engine minimum limits definition.
EC_OFF_PT	scalar	Signifies an inconsistent off design point definition.
EC_DES_ON_OUT	scalar	Signifies an inconsistent desired on design output definition.
EC_DES_OFF_OUT	scalar	Signifies an inconsistent desired off design output definition.
EC_AC_DRAG_FILE	scalar	Signifies an inconsistent aircraft drag file.
EC_AC_CONSTANTS	scalar	Signifies an inconsistent aircraft constants definition.
EC_AC_TO_WEIGHT	scalar	Signifies an inconsistent aircraft gross takeoff weight definition.
EC_AC_VAR_SIZE	scalar	Signifies an inconsistent scaleable aircraft definition.
EC_MISS_PROFILE	scalar	Signifies an inconsistent mission profile definition.
EC_MISS_CONSTANTS	scalar	Signifies an inconsistent mission constants definition.
EC_INST_LOSS	scalar	Signifies an inconsistent installation loss model definition.
EC_DES_VAR	scalar	Signifies an inconsistent design variable definition.
EC_DES_OBJ_WIM	scalar	Signifies an inconsistent design objectives definition for cases involving an aircraft mission.

Global Data Created by Init Errors - continued

Name	*Data Type*	*Brief Description*
EC_DES_OBJ_WOM	scalar	Signifies an inconsistent design objectives definition for cases not involving an aircraft mission.
EC_GA_OPTS	scalar	Signifies an inconsistent GA options definition.
E_INTERACTION	scalar	Signifies a data element interaction conflict.
EI_SACA	scalar	Signifies a stand alone cycle analysis interaction conflict.
EI_SAMA	scalar	Signifies a stand alone mission analysis interaction conflict.
EI_EOWIM_FAC	scalar	Signifies an engine optimization with mission for fixed aircraft interaction conflict.
EI_EOWIM_SAC	scalar	Signifies an engine optimization with mission for scaleable aircraft interaction conflict.
EI_EOWOM	scalar	Signifies an engine optimization without mission interaction conflict.

4.11 Init Miss Constants

With the exception of commonly used TERMAP PIC codes, the global data defined by this unit is given in Table 4.8.

Table 4.8: Global Data Created by Init Miss Constants

Name	*Data Type*	*Brief Description*
C_THRUST_MODE	scalar	MODE value for TERMAP thrust throttling operation.
C_T_5_MODE	scalar	MODE value for TERMAP non-thrust throttling operation.
C_NO_USE_AB	scalar	Value for permitting afterburner use.
C_MAY_USE_AB	scalar	Value for forbidding afterburner use.
C_NUM_TERMAP_CALLS	scalar	Number of TERMAP calls.
C_NUM_TERMAP_CRSHS	scalar	Number of TERMAP crashes.
C_YES	scalar	Value for yes.
C_NO	scalar	Value for no.
C_LI_NOT_POSS	scalar	Linear interpolation not possible.
C_LI_POSS	scalar	Linear interpolation possible.
C_ATM_TABLE	matrix	Atmospheric data table.
C_ATMIDX_ALT	scalar	Index into atmospheric table at which altitude is located.
C_ATMIDX_TEMP	scalar	Index into atmospheric table at which temperature is located.

Global Data Created by Init Miss Constants - continued

Name	*Data Type*	*Brief Description*
C_ATMIDX_PRES	scalar	Index into atmospheric table at which pressure is located.
C_ATMIDX_DELT	scalar	Index into atmospheric table at which delta is located.
C_ATMIDX_DENS	scalar	Index into atmospheric table at which density is located.
C_ATMIDX_SSPD	scalar	Index into atmospheric table at which sound speed is located.
C_ATMIDX_DELG	scalar	Index into atmospheric table at which delta gamma product is located.
C_DRGIDX_MACH	scalar	Index into A/C drag table at which mach is located.
C_DRGIDX_K1	scalar	Index into A/C drag table at which k1 is located.
C_DRGIDX_K2	scalar	Index into A/C drag table at which k2 is located.
C_DRGIDX_CDO	scalar	Index into A/C drag table at which cdo is located.
C_DRGIDX_MSCK	scalar	Index into A/C drag table at which msck is located.
C_GRAV_ACCEL	scalar	Acceleration of gravity.
C_PI	scalar	π
C_DEG_TO_RAD	scalar	Degrees to radians.
C_RAD_TO_DEG	scalar	Radians to degrees.
C_MIN_TO_SEC	scalar	Minutes to seconds.
C_HR_TO_SEC	scalar	Hours to seconds.
C_SEC_TO_HR	scalar	Seconds to hours.
C_MIN_TO_SEC	scalar	Minutes to seconds.
C_SEC_TO_MIN	scalar	Seconds to minutes.
C_NM_TO_FT	scalar	Nautical miles to feet.
C_FT_TO_NM	scalar	Feet to nautical miles.
C_SL_SSPD	scalar	Sea level sound speed.
C_SL_PRESS	scalar	Sea level pressure.
C_ALLOW_LEG_FAIL	scalar	Allow a leg failure.
C_NO_ALLOW_LEG_FAIL	scalar	Do not allow a leg failure.
C_LF_CSC_LEG	scalar	Load factor for a constant speed climb leg.
C_LF_HA_LEG	scalar	Load factor for a horizontal acceleration leg.
C_LF_CA_LEG	scalar	Load factor for a climb and acceleration leg.
C_LF_TO_LEG	scalar	Load factor for a takeoff leg.
C_LF_CASC_LEG	scalar	Load factor for a constant altitude speed cruise leg.

Global Data Created by Init Miss Constants - continued

Name	*Data Type*	*Brief Description*
C_LF_BCMBCA_LEG	scalar	Load factor for a best cruise mach / best cruise altitude leg.
C_LF_LOIT_LEG	scalar	Load factor for a loiter leg.
C_LF_CEHM_LEG	scalar	Load factor for a constant energy height maneuver leg.
C_OFFIDX_IFAIL	scalar	Index into off design data at which IFAIL is located.
C_OFFIDX_SFC	scalar	Index into off design data at which SFC is located.
C_OFFIDX_TSFC	scalar	Index into off design data at which TSFC is located.
C_OFFIDX_UN_THRUST	scalar	Index into off design data at which uninstalled thrust is located.
C_OFFIDX_THRUST	scalar	Index into off design data at which installed thrust is located.
C_OFFIDX_INLET_LC	scalar	Index into off design data at which the inlet loss coefficient is located.
C_OFFIDX_NOZZLE_LC	scalar	Index into off design data at which the nozzle loss coefficient is located.
C_OFFIDX_MODE	scalar	Index into off design data at which MODE is located.
C_OFFIDX_PLA	scalar	Index into off design data at which PLA is located.
C_OFFIDX_PCTRH_1	scalar	Index into off design data at which PCTRH(1) is located.
C_ZERO_MACH	scalar	A zero mach number.
C_NULL_RETURN	scalar	A null return value.
C_PASS	scalar	Pass.
C_FAIL	scalar	Fail.
E_ABORT_MISS	scalar	Abort mission.
E_NO_ERROR_IN_LEG	scalar	No error in leg.
E_NO_WARNG_IN_LEG	scalar	No warning in leg.
E_BAD_SWITCH_NUM	scalar	Bad switch number error.
E_LIN_INT_ERROR	scalar	Linear interpolation error.
E_LIFT_VIOLATE	scalar	Lift coefficient violation.
E_INSUFF_THRUST	scalar	Insufficient thrust error.
E_SWITCH_CONST	scalar	Switch constraint error.
E_BAD_AB_OPTION	scalar	Bad afterburner option error.
E_TO_DIST	scalar	Takeoff distance error.
E_LEG_FAIL_CONT	scalar	Leg failure continutation error.
E_THROTTLE_AB	scalar	Afterburner throttling error.

Global Data Created by Init Miss Constants - continued

Name	*Data Type*	*Brief Description*
E_TERMAP_CRASH	scalar	TERMAP crash error.
E_TERMAP_INFEAS	scalar	TERMAP infeasible error.
E_UNEXPECTED	scalar	Unexpected error.
E_REQ_UNIN	scalar	Required uninstalled data error.
E_LOSS_FAIL	scalar	Installation loss model failure.

4.12 Init Kriging Data

The global data defined by this unit is given in Table 4.9.

Table 4.9: Global Data Created by Init Leg Mapping

Name	*Data Type*	*Brief Description*
K_USE_KRIGING	scalar	Reference number to use kriging.
K_NO_USE_KRIGING	scalar	Reference number to not use kriging.
K_GOOD_ESTIMATE	scalar	Reference number for a good kriged estimate.
K_BAD_ESTIMATE	scalar	Reference number for a bad kriged estimate.
K_QUALITY_MEASURE	scalar	Quality measure option for kriging code.
K_KRIGING_TOL	scalar	Kriging tolerance to be used with quality measure.